CMPE 483 Project 1 Report

ADALET VEYİS TURGUT 2017400210

ERENCAN UYSAL 2017400069

MUHAMMED GÖKTEPE 2017400162

In this project, we are asked to implement an autonomous decentralized lottery. Each lottery lasts one week and it consists of two stages. Submission stage lasts four days and users buy tickets in this stage. In reveal stage users approve and reveal their tickets. Successfully revealed tickets have a chance to win the lottery but the others have lost their chance and can be refunded.

We have four different contracts in this project. At the beginning, we created our token which is Turkish Lira (TL). This token will be used while buying tickets in this lottery. It has three functions to mint, transfer money and approve the maximum amount that can be transferred. At the second contract we implemented ticket NFT's. It inherits the ERC721 token contract. It used counters to assign token ids to each ticket. At the third contract we implemented the necessary functions for the lottery. This contract needs the address of the deployed TL contract. Then we implemented the user contract. It needs the address of the deployed lottery and TL contract. In this contract users can deposit money to their account and transfer it to the lottery contract to buy tickets. Also, users can hash random numbers that they used while buying the tickets.

1. TL.sol

1.1. What it does?

This contract represents the money that was used in the lottery and inherits ERC20 class. We coded three extra methods namely *giveMoney(), myTransfer()* and *myApprove()* to this contract since delegatecall was not working from lottery.sol contract.

giveMoney() function is called to give users some TL initially. So that they can buy tickets with it. This function is the part of the initialization phase.

myTransfer() function is not used. We firstly implemented it to transfer without delegatecall, but after implementing *myApprove* function, we did not use this.

myApprove() functions gives lottery contract allowance to spend money from callers account so that it can deposit TL.

1.2. Global Variables

None

1.3. Functions

1.3.1. function giveMoney(address _to, uint256 _amnt) public payable returns (bool)

1.3.1.1. Parameters

1.3.1.1.1. _to: address to assign the newly minted TL.

1.3.1.1.2. _amnt: amount of TL to be minted.

1.3.1.2. What It Does?

This function mints given amount of TL and assigns to the address given.

1.3.2. function myTransfer(address _from, address _to, uint256 _amnt) public payable returns (bool)

1.3.2.1. Parameters

1.3.2.1.1. _from: address that TL withdrawn from.

1.3.2.1.2. _to: address to deposit the TL.

1.3.2.1.3. _amnt: amount of TL to be transferred.

1.3.2.2. What It Does?

1.3.3. function myApprove(address _origin, address _spender, uint256 _amnt) public returns (bool)

1.3.3.1. Parameters

1.3.3.1.1. _origin: address of the owner of the account that gives allowance.

1.3.3.1.2. _spender: address of the spender of TL that takes allowance.

1.3.3.1.3. _amnt: amount of TL that allowed for further transfers.

1.3.3.2. What It Does?

Since the *transferFrom()* function requires an allowance of the address that money deposited from, this function helps users to give an allowance to the lottery contract.

2. Ticket.sol

2.1. What it does?

This contract represents the ticket that was sold in the lottery and inherits ERC721 class. We coded one extra method namely createTicket() to this contract. This function is called when a new ticket is bought and a newly created ticket is assigned to the buyer. This contract also uses a variable from Counters class. This counter starts from 0 and is incremented after every ticket creation. This counter is ticket no. This contract is deployed in the constructor of the lottery contract.

2.2. Global Variables

2.2.1. Counters.Counter private _tokenIds : This variable starts from 0 and increments with each newly created ticket. This is the ticket number the user sees externally. Since this is an NFT, each ticket is unique and it's not divisible.

2.3. Functions

2.3.1. function createTicket(address _owner) public returns (uint256)

2.3.1.1. Parameters

2.3.1.1.1. _owner: This parameter is the address of the owner of this ticket.

2.3.1.2. What It Does?

This function calls ***_mint()*** function of the ERC721 class with _owner parameter and _tokenIds variable. After calling ***_mint()*** function, it increments the counter by one.

3. Lottery.sol

3.1. What it does?

In this contract we implemented all requested functions for an autonomous lottery.

3.2. Global Variables

3.2.1. Status: Enumaration for status field of the tickets.

3.2.2. status_of_tickets: Stores tickets status. If ticket won the lottery it's status changed to WINNER. If the ticket is correctly revealed REVEALED. If not CANCELLED. If the reward is collected COLLECTED. If the ticket refunded REFUNDED.

3.2.3. amount_of_tickets: Stores the amount of prize each ticket won.

3.2.4. MAX_INT: Maximum integer value.

3.2.5. balances: Stores the balances of users. Maps addresses to uint balances.

3.2.6. ticket_hash_map: Stores the hashed random numbers provided by the user in submission stage for each ticket.

3.2.7. hashed_tickets: Stores the hashes of tickets in each lottery. Maps lottery to hashed tickets.

3.2.8. winner_tickets: Stores winner ticket numbers in each lottery.

3.2.9. ticket_nfts_of_user: user => [lottery no => [ticket nos]] Stores the ticket numbers in each lottery owned by each user.

3.2.10. buyed_tickets: Stores bought tickets in each lottery.

3.2.11. refund_tickets: Stores tickets refund condition.

3.2.12. revealed_rnds: Stores the correctly provided random numbers in reveal stage.

3.2.13. lastCalculatedLotteryNo: Stores last calculated lottery number to check if a lottery ended.

3.2.14. revealed_tickets: Stores the ticket numbers of correctly revealed tickets in reveal stage.

3.2.15. purchaseInterval: Purchase time in seconds.
3.2.16. revealInterval: Reveal time in seconds.
3.2.17. tl_contract: Address of TL contract.
3.2.18. ticket_contract: Address of ticket contract.
3.2.19. initialLotteryTime: Stores the beginning of first lottery in unix time.

3.3. Functions

3.3.1. function depositTL(uint _amnt) public payable

3.3.1.1. Parameters

3.3.1.1.1. _amnt: Amount of money that will be deposited to user account

3.3.1.2. What It Does?

First we check overflow and underflow conditions. Then we transfer TL tokens from caller to this contract. Finally we increase the balance of the caller

3.3.2. function withdrawTL(uint _amnt) public payable

3.3.2.1. Parameters

3.3.2.1.1. _amnt: Amount of money that will be withdraw from user account

3.3.2.2. What It Does?

First we check overflow and underflow conditions. Then we transfer TL tokens from this contract to the caller. Finally we decrease the balance of the caller.

3.3.3. function buyTicket(bytes32 hash_rnd_number) public payable

3.3.3.1. Parameters

3.3.3.1.1. hash_rnd_number: bytes32 value provided by the user while buying the ticket in submission stage.

3.3.3.2. What It Does?

User can buy tickets usig this function. We must be in submission stage! We check whether caller has enough money to buy a ticket. We check if this bytes32 number submitted previously, if yes revert. We withdraw the ticket cost from callers balance. We push this hash_rnd_number to hashed_tickets array with current lottery number. We create and assign a new ticket to the user. We push this ticket no to our buyed_tickets array with current lottery id. We create a mapping between ticket no and submitted hash rnd number. We assign ticket to user with current lottery id.

3.3.4. function collectTicketRefund(uint ticket_no) public payable

3.3.4.1. Parameters

3.3.4.1.1. ticket_no: ticket no of the ticket that owned by the user

3.3.4.2. What It Does?

Users can get half of the ticket money back if previously submitted random numbers are not submitted correctly in the reveal stage. We update the balances and change the status of ticket to REFUNDED.

3.3.5. function revealRndNumber(uint _ticket_no, uint _N) public

3.3.5.1. Parameters

3.3.5.1.1. _ticket_no: ticket no of the ticket that owned by the user

3.3.5.1.2. _N: Random number that user provides.

3.3.5.2. What It Does?

Users reveals their tickets in reveal stage. They provide the random number that they used while buying the ticket. If it isn't matches with the previously provided number than winning chance of the ticket is lost. We take the hash of the number _N as user did in submission stage. Store this ticket in revealed_tickets array and associate it with current ticket no.

3.3.6. function indexOf(uint256[] memory _arr, uint256 _elem) private returns (int)

3.3.6.1. Parameters

3.3.6.1.1. _arr: Array that will be searched.

3.3.6.1.2. _elem: Element that is searched in the array.

3.3.6.2. What It Does?

Helper function. It returns the index of _elem in given _arr.

3.3.7. function hashIndex(bytes32 _hash) private returns(int)

3.3.7.1. Parameters

3.3.7.1.1. _hash: Hash of the submitted random number in submission stage.

3.3.7.2. What It Does?

Helper function, returns the index of _hash in global array hashed_tickets[current_lottery_no].

3.3.8. function inRevealStage() private view returns (bool)

3.3.8.1. Parameters

3.3.8.1.1. None

3.3.8.2. What It Does?

Helper function, returns whether we are in reveal stage.

3.3.9. function myGetLotteryNo(uint unixtimeinweek) public returns (uint lottery_no)

3.3.9.1. Parameters

3.3.9.1.1. unixtimeinweek: Current time in unix.

3.3.9.2. What It Does?

It calculates current lottery number. If it is not matches with previously calculated lottery number which means a lottery ended and new one begins than it calculates winner tickets and resets variables and starts the ne lottery.

3.3.10. function getLotteryNo(uint unixtimeinweek) public view returns (uint lottery_no)

3.3.10.1. Parameters

3.3.10.1.1. unixtimeinweek: Current time in unix.

3.3.10.2. What It Does?

It calculates current lottery number.

3.3.11. function isFinished(uint lottery_no) private view returns (bool)

3.3.11.1. Parameters

3.3.11.1.1. lottery_no: This is the lottery number.

3.3.11.2. What It Does?

It checks whether provided lottery is ended or not.

3.3.12. function getLastOwnedTicketNo(uint lottery_no) public view returns(uint,uint8 status)

3.3.12.1. Parameters

3.3.12.1.1. lottery_no: This is the lottery number.

3.3.12.2. What It Does?

User provides a lottery number to learn the last ticket number that owned by him/her. It returns a ticket number and its status.

3.3.13. function getIthOwnedTicketNo(uint i,uint lottery_no) public view returns(uint,uint8 status)

3.3.13.1. Parameters

3.3.13.1.1. i: i'th ticket that bought

3.3.13.1.2. lottery_no: This is the lottery number.

3.3.13.2. What It Does?

User provides a lottery number, and an integer i to learn the i'th ticket number that owned by him/her. It returns a ticket number and its status.

3.3.14. function checkIfTicketWon(uint ticket_no) public view returns (uint amount)

3.3.14.1. Parameters

3.3.14.1.1. ticket_no: ticket no of the ticket that owned by the user

3.3.14.2. What It Does?

It returns the prize amount won by the ticket.

3.3.15. function collectTicketPrize(uint ticket_no) public

3.3.15.1. Parameters

3.3.15.1.1. ticket_no: ticket no of the ticket that owned by the user

3.3.15.2. What It Does?

Checks whether the user calling the function is the owner of the ticket. If he is the owner of the ticket and the ticket has won an award, it transfers the prize amount to the user's address.

3.3.16. function findWinners() private

3.3.16.1. Parameters

3.3.16.1.1. None

3.3.16.2. What It Does?

Calculates the winner tickets. It takes the revealed tickets random number provided by the users and makes bitwise xor operation. Then it calculates the total amount of money collected in this lottery and how many tickets will win prizes. Lastly it finds the winning ticket and hashes the N_xor value and repeats these steps until it finds all winner tickets. It changes the status of the winning tickets to WINNER.

3.3.17. function getIthWinningTicket(uint i, uint lottery_no) public view returns (uint ticket_no,uint amount)

3.3.17.1. Parameters

3.3.17.1.1. i: i'th ticket that win

3.3.17.1.2. lottery_no: This is the lottery number.

3.3.17.2. What It Does?

Finds i'th winning ticket in the provided lottery and returns the prize amount of that ticket.

3.3.18. function getTotalLotteryMoneyCollected(uint lottery_no) public view returns (uint amount)

3.3.18.1. Parameters

3.3.18.1.1. lottery_no: This is the lottery number.

3.3.18.2. What It Does?

Finds total collected money in the provided lottery.

4. User.sol

4.1. What it does?

This contract is a helper to the lottery contract. With this contract, users can give themselves some money to buy tickets and approve our lottery contract to spend money for themselves. Also they can hash numbers before buying a ticket. This contract needs the addresses of the TL and lottery contracts to be deployed.

4.2. Global Variables

4.2.1. MAX_INT: constant maximum value check overflow conditions

4.2.2. lottery_contract: Variable that holds lottery contract. Its address is given in the constructor and initialized there.

4.2.3. tl_contract: Variable that holds TL contract. Its address is given in the constructor and initialized there.

4.3. Functions

4.3.1. function giveMoney(uint _amnt) public payable

4.3.1.1. Parameters

4.3.1.1.1. _amnt: Amount of TL to give the caller.

4.3.1.2. What It Does?

It calls the giveMoney*()* function of the TL contract with msg.sender address.

4.3.2. function approveContract(uint _amnt) public

4.3.2.1. Parameters

4.3.2.1.1. _amnt: Amount of TL that user allows the lottery contract to spend.

4.3.2.2. What It Does?

It calls *myApprove()* function of the TL contract with msg.sender address.

4.3.3. function hashNumber(uint256 _num) public returns (bytes32)

4.3.3.1. Parameters

4.3.3.1.1. _num: Integer to be hashed.

4.3.3.2. What It Does?

It hashes the given _num with msg.sender address and returns it.

Average gas usages: [5 - 10 - 100 - 200] accounts

function depositTL(uint amnt) public :

72400 70700 69170 68870

function withdrawTL(uint amnt) public:

61191 61191 61191 61191

function buyTicket(bytes32 hash_rnd_number) public:

239168 197546 202377 207208

function collectTicketRefund(uint ticket_no) public:

41438 40365 41344 40344

function revealRndNumber(uint ticketno, uint rnd_number) public:

100285 101417 115067 108907

function getLastOwnedTicketNo(uint lottery_no) public view returns(uint,uint8 status):

30005 30005 30005 30005

function getIthOwnedTicketNo(uint i,uint lottery_no) public view returns(uint,uint8 status):
22213 24381 23245 24563

function checkIfTicketWon(uint ticket_no) public view returns (uint amount):

24459 24447 24465 24438

function collectTicketPrize(uint ticket_no) public:

41530 52195 49135 51125

function getIthWinningTicket(uint i, uint lottery_no) public view returns (uint ticket_no,uint amount):

29535 29535 29535 29535

function getLotteryNo(uint unixtimeinweek) public view returns (uint lottery_no):

28370 28370 28370 28370

function getTotalLotteryMoneyCollected(uint lottery_no) public view returns (uint amount):

24257 26263 26645 25344

TEST RESULTS:

We wrote a test script using hardhat. In this script, we tested our contracts with 5, 10, 100, 200 and 400 users. We first deployed contracts, then created users. Afterwards, we gave initial TL to each user, then deposited these TL's. After depositing money we bought one ticket for each user. Then we called the get last owned ticket function.
After buying tickets in the submission stage, we forward time. Then reveal random numbers, but make intentional mistakes in some of them.
After revealing numbers, we again forward the time for 3 days so that the reveal period ends. Then we check whether the ticket won a prize and collect it.
Finally, we print out average gas usages for the functions above.

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.			
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.			