**hatıp kabak**

**2016705027**

***Lottery assıgnment cmpe483-blockchaın programmıng***

INTRODUCTION

This assignment is a decentralised lottery contract that is powered

purely by the Ethereum blockchain. The lottery is completely decentralized. Therefore,

there is no need to trust a centralized authority. In this lottery, each participant

generates their three random numbers and then creates their commitment by hashing.

These are supplied to system by the purchaser. After sending these to the contract and

ending the submissions, each participant sends their random numbers to the contract.

These numbers must be appended to their address. Our contract approves that the

keccak256(append(firstNumber, msg.sender)) matches the original submission. If

participant does not supply valid three numbers in a valid time (lottery round is 20000),

he/she cannot be won the lottery. All of the numbers belong to participants expose to

XOR operation. XOR operation are performed amongst first numbers of users. It is done

for second numbers and third numbers in the same way.

At the end of the round, we have three result numbers after finishing XOR operations.

These result numbers is used to determine the first prize, second prize and third prize

winners by calculating index with “resultNumber%numberOfVerifiedUsers”. Winners

can withdraw their money whenever they want.

ENVIRONMENT

* Truffle 4.1.17
* Ganache CLI
* Mocha

IMPLEMENTATION OF THE CONTRACT

A)DATA TYPES

A.1)STRUCTS

A.1.2)Player

The Player struct represents one participation. One adress can have multiple players,that is, one address can purchase more than one tickets. The Player struct contains 5 struct variables.

* firstHashValue : hash of first random number
* secondHashValue : hash of second random number
* thirdHashValue: hash of third random number
* ticketInfo: ticket form such as 1 represents full ticket
* blockNumber: block number when ticket is bought
* lotteryHalfRoundIndex: It shows lottery half round index (index of period is 20000,for ex:submission stage is a half round of one lottery, reveal stage is another round of one lottery )

A.1.2) PlayerRevealedInfo

The PlayerRevealedInfo represents participants that their random

numbers matches hash of the these numbers. That is, these participants are entitled to participate in the draw.

* entitledToDrawAddres: address of entitled to participate in the draw
* ticketInfo: ticket form such as 1 represents full ticket
* revealedHalfRoundIndex: shows lottery half round index (index period of 20000)

A.2) ARRAY OF STRUCTS

A.2.1) PlayerRevealedInfo[] public revealedList;

list of participants that are entitled to participate in the draw

A.3) MAPPINGS

A mapping is referred to a hash table, which consists of key types and value type pairs.

A.3.1) mapping(address => Player[]) players;

This allows us look up all players with the Ethereum address and retrieve all hash values of random numbers, ticket info and block number.

A.3.2) mapping(address => uint256) prizeMap;

This allows the winners with the Ethereum address withdraw their prizes

A.4) OTHER DATA TYPES

A.4.1) XORFirstNumber

XOR operation result number of the first numbers

A.4.2) XORSecondNumber

XOR operation result number of the second numbers

A.4.3) XORThirdNumber

XOR operation result number of the third numbers

A.4.4) fullTicket

Shows full ticket

A.4.5) halfTicket

Shows half ticket

A.4.6) quarterTicket

Shows quarter ticket

A.4.7) balance

Represents circulating money in the lottery

A.4.8) firstBlockNumber

Represents firstBlockNumber when initializing lottery

B)FUNCTIONS

B.1)constructor-> “function Lottery()”

The field balance sets to 0 and firstBlockNumber sets to block.number at the creation time

B.2)buyTicket(firstHashValue, secondHashValue, thirdHashValue)

In this function, participant send hash of three random numbers (numbers appends to participant’s address for each number) to the contract. According to money that participant sends, ticket form is determined and submission is fired.

If ticket fee (msg.value) is greater than 2 finney or equals to 2 finney and less than 4 finney, the ticket form is quarter. If ticket fee (msg.value) is greater than 4 finney or equals to 4 finney and less than 8 finney, the ticket form is half.

If ticket fee (msg.value) is greater than 8 or equal to 8, the ticket form is full.

After buying ticket, submission function fires.In addition to this, it is looked at whether drawing conditions meet or not. If it is draw function also fires

B.3) submission(firstHashValue, secondHashValue, thirdHashValue, ticketInfo)

In this function, participant info is pushed into the mapping players in terms of participant address. Then, money that participant sends added to the lottery balance.

B.4) revealStage(firstNumber,secondNumber, thirdNumber)

During the reveal stage phase, the participants must send in their secret integers, which are hashed and compared to their original submission.

We look at whether the random numbers are completely revealed or not.

If it is, participant address that is entitled to participate in the draw is added to revealed list together with the ticket info(PlayerRevealedInfo). All of the three numbers are expose to XOR operation. First numbers of participants are exposed to XOR operation amongst themselves. It is applied to second and third numbers in the same way.

In the end, participant is deleted from players. This operation prevents that same ticket wins the lottery more than one.

In this stage, block.number must be greater than 20000.

B.5) draw()

Winner is determined in this function, after all secret numbers have been successfully collected. One lottery runs for a period of 40000 blocks. Ticket purchase and random number submission stage runs for a period of 20000 blocks like in the reveal stage. There is no any reveal process first 20000 blocks. After taking all of them into consideration, draw is performed if block.number is greater than 20000(or equal) and block.number *modulus* 20000 is equal to 0 and revealedList.length (numberOfVerifiedUsers) is greater than 0.

First prize,second prize and third prize are determined ,respectively XORFirstNumber, XORSecondNumber and XORThirdNumber. These XORs is divided by revealedList.length to determine the index of winner in the list. (For ex: revealedList[XORFirstNumber%revealedList.length])

After determining for each prize, prize of the unique address added to prizeMap(like an account) and subtracted to lottery balance. Thus, the remain balance is also used for next lottery. (current lottery round plus the amount that was carried over to the current round from the previous round.)

We also used removeWinnerFromRevealedList function to prevent that same ticket win the lottery more than one.

In the end of the function, we delete the revealedList for the next lottery.

C)HELPER FUNCTIONS

C.1) isDrawPossible()

This function determines whether drawing performs or not.

C.2) revealSecretNumberHash(firstNumber, secondNumber, thirdNumber,index)

This function reveals whether hash of appending number and address matches hash that was submitted in the first stage. keccak256 is used to perform hash operation. Sha3 is deprecated.

C.3)getKeccakHash(num)

This sends the hash by appending number with msg.sender.

C.4) addRevealedInfoToList(ticketInfo)

This function is used to add verified user into the revealedList

C.5) getLotteryHalfRoundIndex ()

It sends the index of every stage.

C.6) performXOROperations(num,num,num)

This function perform XOR operation for every three numbers supplied by purchaser

C.7) deletePlayerFromPlayers(index)

This prevents that same ticket wins the lottery more than one.

C.8) removeWinnerFromRevealedList(winnerAddress, ticketInfo)

It prevents that same ticket wins the lottery more than one.

C.9) firstPrizeForTicketInfo(ticketInfo)

It determines first prize in terms of ticket info

C.10) secondPrizeForTicketInfo (ticketInfo)

It determines second prize in terms of ticket info

C.11) thirdPrizeForTicketInfo (ticketInfo)

It determines third prize in terms of ticket info

C.12) withdraw()

The participant that wins the lottery withdraw the money from prizeMap whenever he/she via this function

D) SHOWING CORRECTNESS OF THE IMPLEMENTATION

* Reveal starts after reaching block number 20000.

I add a check for this in revealStage function

(block.number-firstBlockNumber>=20000)

* First lottery draw starts after reaching block number 40000. Then, every 20000 block should be possible to draw. Finally, revealedList, that is purchasers that is able to participate in lottery, should be greater than 0.

I add a check for these conditions in isDrawPossible function

(block.number-firstBlockNumber)>=40000 && (block.number-firstBlockNumber)%20000==0 && revealedList.length>0

Lottery1-end:20000 Lottery2-end:40000 Lottery3-end:60000 ….

* The lottery should employ submission and reveal. .isDrawPossible function checks every call of buyTicket and revealStage functions. Therefore our lottery is fully autonomous.
* When a purchaser buys a ticket, if it is possible, purchaser’s info submit with Player Struct in players mapping. Then purchaser can reveal his/her numbers.
* When a player performing reveal stage and if the player is able to participate in draw, the player added to revealedList and deleted from mapping players. This prevents that same ticket wins the lottery more than one.

deletePlayerFromPlayers(index);

* After drawing, deleting revealedList is performed

delete revealedList;

* Our hashing mechanism is performing by appending number with the address.That is, user should send hash of appending the number and the address when buying ticket.
* Submission and reveal stage in a lottery happens successively.For example:

STAGE BLOCKS

Submission-1 0-20000

Reveal-1 20000-40000

Submission-2 20000-40000

Reveal-2 40000-60000

Submission-3 40000-60000

Reveal-3 60000-80000

Reveal-3 have to look at only Submission-3 or Reveal-2 have to look at only Submission-2 or vice versa. To manage this, I assign an index to every stage for submissions and reveal. Every 20000 index is increased by 1.Difference indexes of between submission and reveal in one lottery is always 1.

As a result , If previously submitted random numbers are not submitted correctly in the reveal stage, the chance of winning is lost. This can be done by looking at indexes of stages. One reveal stage can look at the submission stage in the same lottery.

Difference indexes of Submissions and Reveals in the SAME LOTTERY is always 1.

STAGE BLOCKS INDEX

Submission-1 0-20000 0🡪LOTTERY1

Reveal-1 20000-40000 1🡪LOTTERY1

Submission-2 20000-40000 1🡪LOTTERY2

Reveal-2 40000-60000 2🡪LOTTERY2

Submission-3 40000-60000 2🡪LOTTERY3

Reveal-3 60000-80000 3🡪LOTTERY3

* Ticket refund is not possible
* Total lottery balance should be money collected from sale of tickets at the current lottery round plus the amount that was carried over to the current round from previously rounds. In order to manage this, balance variable is separately kept. All the money that comes from tickets whenever ticket buys is added to balance variable. All the prizes are subtracted from this variable.
* A winning user should be able to withdraw his prize anytime after the lottery round ends. While draw function is performing, prize of winners added to prizeMap together with their address. Therefore, after drawing ends, winners can withdraw their total prizes whenever they want.
* With removeWinnerFromRevealedList function, it is prevented that same address can win the prize in same lottery
* A purchaser can buy only one ticket at a time.