COMP6451 T1 2022

Assignment 2

Haozhe Chen

z5142012

**Part 1: Contract structure overview**

This assignment can be divided into two parts of contract:

1. Share contract used ERC20 standard interface
2. 图示

   描述已自动生成Fundraising contract which is used to bid and revel.

**[Round 0] Initial**

In this round, beneficiary (aka NeverPay company), will deploy the fundraising contract (NeverPayFundraising.sol) into the blockchain. In the fundraising contract’s constructor, the share contract (ERC20NeverPayToken.sol) will be initiated.

Before this round end, the fundraising contract should keep 10000 tokens.

**[Round 1] Bid**

In this round, investors are able to call bid function to make a blinded bid. The parameter should be a hash value = keccak256(share, value, nonce).

**[Round 2] Reveal**

In this round, investor will call reveal function to open their bid in round1 and pay to the contract. The parameters are shares, value and nonce.

If hash value of these three parameters is equal to bid record in round1 and amount paid is larger than price \* shares, this reveal is successful. Otherwise, this reveal is failed.

**[After Round 2] Issue**

This round is used to let every investor to get their shares and refunds. Beneficiary can also call this function to get ETH collected during the last round (only successful bid).

The issue function will firstly sort every bid (reveal successfully in round2), then the successful bid’s owner will get the shares, failed bid’s owner will get the refunds.

**Part 2a: Data model (Token contract)**

|  |  |  |
| --- | --- | --- |
| **Token contract: ERC20NeverPayToken.sol** | | |
| **Property** | **Type** | **Description** |
| name | uint256 | Token’s name |
| decimals | uint8 | How many decimals to show |
| symbol | string | Token’s identifier |
| balances | mapping  address => uint256 | user\_adress => balances  User’s balances |
| totalSupply | uint | Token’s total supply |

The token contract is followed by ERC20 token interface. Every users’ balance is stored by a mapping structure explicitly.

|  |  |  |
| --- | --- | --- |
| **Function: transfer** | | |
| **Parameters** | **Type** | **Description** |
| \_to | address | Receiver’s address |
| \_value | uint256 | Transfer amount |
| **Returntype** | **Description** | |
| bool | Transfer successfully or failed | |

This function is used to let sender to transfer money to certain address.

|  |  |  |
| --- | --- | --- |
| **Function: transferFrom** | | |
| **Parameters** | **Type** | **Description** |
| \_from | address | Sender’s address |
| \_to | address | Receiver’s address |
| \_value | uint256 | Transfer amount |
| **Returntype** | **Description** | |
| bool | Transfer successfully or failed | |

This function is also used to transfer money. However, this function should be called by receiver to initiate a transaction from a certain address. Which also need sender’s to approve this transaction with approve function.

|  |  |  |
| --- | --- | --- |
| **Function: approve** | | |
| **Parameters** | **Type** | **Description** |
| \_spender | address | Receiver’s address |
| \_value | uint256 | Approve transfer amount |
| **Returntype** | **Description** | |
| bool | Approve successfully or failed | |

This function should be called by sender to approve amount of money transferred by certain address. In case of receiver ‘stole’ money from sender (can initiate transfer with any amount from any address).

|  |  |  |
| --- | --- | --- |
| **Function: balanceOf** | | |
| **Parameters** | **Type** | **Description** |
| \_owner | address | Owner’s address |
| **Returntype** | **Description** | |
| uint256 | Owner’s balance | |

This function is used to get the balance of certain address.

|  |  |  |
| --- | --- | --- |
| **Function: allowance** | | |
| **Parameters** | **Type** | **Description** |
| \_owner | address | Sender’s address |
| \_spender | address | Receiver’s address |
| **Returntype** | **Description** | |
| uint256 | Value approved by sender to transfer. | |

This function is used to check the amount of value approved by sender to transferred to receiver.

**Part 2b: Data model (Fundraising contract)**

|  |  |  |
| --- | --- | --- |
| **Token contract: NeverPayFundraising.sol** | | |
| **Property** | **Type** | **Description** |
| token | ERC20NeverPayToken | Token’s contract address. |
| beneficiary | address payable | Beneficiary’s address. |
| bidEnd | uint | timestamp of round1 end:  22/04/2022 |
| revealEnd | uint | timestamp of round2 end:  27/04/2022 |
| bids | mapping(address => mapping(bytes32 => bool)) | addr => (bid\_hash\_value => valid)  Whether this bid is able to reveal. |
| refunds | mapping(address => uint) | addr => refund\_amount  Record refund amount of every investor. |
| validBid | struct:  address addr  uint shares  uint price  uint bid\_order | Structure of a valid bid.  Contain bidder’s address, bid’s share and price, and order of this bid. |
| validBids | validBid[] | List contain every valid bid. |
| bidOrder | mapping(bytes32 => uint) | bid\_hash\_value => order  Mapping to keep order of each bid. |
| order | uint32 | Record global order of bid. |
| issued | mapping(address => bool) | addr => issued  Mapping to record whether this issue has been issued before. |

This contract is used to fundraise the ETH from investors. Several mapping structures are used to keep bid’s information:

**[bids]** is used to record whether this bid is able to reveal, it locate different bids by their investors’ address and hash\_value. The value is true when bid is been created. When investors withdraw this bid or reveal this bid, it will be marked as false.

**[refunds]** is used to record refund value of every investor at round1 & 2.

E.g., bid withdraw, reveal failed (mismatched bid info, insufficient payment)

**[bidOrder]** is used to keep every bid’s order at round1, so that when investors reveal their bid, contract can get bid’s order at round1 by bid’s hash\_value.

**[issued]** is used to check whether certain investor has been issued before, to prevent the investors issuing multiple times to get extra shares and refund.

After investors revealing their bid, contract will use **validBid** structure to store the bids, it contains every detail information of this bid to help contract initiate shares transfer of ETH refund after round2. The **order** property of **validBid** structure will be used to keep bids’ order when sort if there are multiple bids with same price.

|  |  |  |
| --- | --- | --- |
| **Function: bid** | | |
| **Parameters** | **Type** | **Description** |
| \_bindedBid | bytes32 | h(shares, price, nonce) |

Investors will use this function to create a bid in round1. Since this fundraising contract is a blinded bid contract. User should use a keccak256 hash function to encrypt their bids information, so that other investors cannot get bids’ detail. And in case of bids with same shares and price, hash function also contains a different nonce given by different investors.

|  |  |  |
| --- | --- | --- |
| **Function: withdrawBid** | | |
| **Parameters** | **Type** | **Description** |
| \_bindedBid | bytes32 | h(shares, price, nonce) |

Investors will use function to withdraw a bid. It will modify **bids[msg.sender][\_bindedBid]** to false, so that this bid cannot be revealed in round2.

|  |  |  |
| --- | --- | --- |
| **Function: reveal** | | |
| **Parameters** | **Type** | **Description** |
| \_shares | uint | Shares contained in bid in round1. |
| \_price | uint | Price contained in bid in round1. |
| \_nonce | bytes32 | Nonce used to calculate hash with price and shares. |

During the round2, investors will give the shares, price, and nonce to reveal function. If hash value calculated by three parameters can be founded in **bids** mapping (from round1), this reveal will be counted as a valid reveal and recorded by **validBids** list.

|  |  |  |
| --- | --- | --- |
| **Function: issue** | | |
| **Parameters** | **Type** | **Description** |
|  |  |  |

This function should be called by investors or beneficiary after round2. It will firstly sort the validBids list, so that some potentially successful bids will become invalid bids because total shares may greater than 10000. At the end of this function, investor can get their shares or refunds depends on their bids’ status. Beneficiary can call this function to calculate how much ETH collected by fundraising and get ETH transfer from the contract.