Description and assumptions of the auction protocol

Public data

- minimumDepositValueInWei: can also be considered as the starting value for bidding.
- endBiddingTime
- endRevealingTime
- endFinalizationTime
- secondHighestRevealedBid
- highestRevealedBid
- highestRevealedBidder
- auctionManager: address.
- BiddersToDeposits: mapping (bidder address ➔ deposit value).
- BiddersToHashedBids: mapping (bidder address ➔ hashed bid).
- BiddersToRevealedBids: mapping (bidder address ➔ revealed bid).
- honestBidders: address payable[] array.
- auctionState: enum (Created, InAction, Ended).

Steps

1. Manager deploy contract:
    - He should provide minimumDepositValueInWei, biddingDurationInMinutes, revealingDurationInMinutes and finalizationDurationInMinutes.
    - The manager places a deposit that is bigger than or equal to the minimum deposit value in Wei which he provided.
    - He places a deposit as an incentive to complete the protocol and not disappear in the middle of the process (he will only be able to refund his deposit in finalization time and auction have ended).
    - Note: the manager can abort the auction and refund his deposit as long as no bidder has placed his deposit yet. Only the manager can call the abort function.
2. Bidders place their deposits and bids in bidding time:
    - Bidders can place them only if the auction is in bidding time and has not ended yet.
    - Deposits must exceed minimum deposit value provided by the manager when he deployed the contract.
    - Each bidder can bid only once.
    - Auction manager cannot place a bid.
3. Bidders reveal their bids by sending the nonce and the bid value in revealing time:
    - Bidders can reveal bid only if the auction has not ended and is in revealing time.
    - Each bidder can reveal his bid only once.

- To reveal a bid, the bidder must have placed a hashed bid in the bidding time.
- The hash of the nonce concatenated with the revealed bid must be equal to the hashed bid placed in the bidding time. Hash function used is keccak256.
- Revealed bid is bigger than deposit value.
- Auction manager cannot reveal a bid as he cannot place one.
- Note: with each bid reveal, the contract compute the first and second highest bid and saves the address of the highest bidder (winner). If only one bidder has placed his bid, then he is the highest bidder and the highest bid value is his revealed bid, the second highest bid is equal to zero.

4. The losing bidders refund their placed deposits in finalization time.
    - Bidders can only refund their deposits in finalization time.
    - Refund deposits cannot be called by manager or winner.
    - Each bidder can refund his deposits only once.
    - The contract checks that for a bidder to refund his deposit he must have followed the protocol (bidder place deposit, bidder placed hashed bid, bidder and bidder revealed his bid).
    - The contract transfer the honest bidder deposit value to his address.
    - Bidders to deposit map mas his address to a deposit of zero value (as the contract has transferred his money to his address already).
5. Winner send money with value (second highest bidding value – deposit value) to manager.
    - Winner can only call this function in finalization time.
    - WinnerSendMoneyToManager function can only be called by winner.
    - Winner must send the value of (second highest bidding value – deposit value) to the manager or otherwise the transaction is reverted.
    - Bidders to deposit map mas his address to a deposit of zero value (as the contract has transferred his money to manager's address already).
    - Note: If second highest bidding value is equal to zero then the bidder will send money with value equal to his deposit only.
6. Manager end auction in finalization time.
    - Function can only be called by manager.
    - The contract transfer the manager deposited value to the manager. Note: the manager already got the second highest bidding value in the previous step.
    - Contract balance now is equal to cheaters deposits.
    - The contract balance is now divided (integer division) on the honest bidders (including the winner) and the manager.

- The manager gest his share and the rest of integer division.
- Note: after the auction has ended the auction state will be set to Ended, which means that the contract is only valid for one auction.

7. Participant end auction after finalization time in case the manager didn't finalize the auction.
    - Function cannot be called by manager, is called after finalization time and auction is not in ended state.
    - Can only be called by an honest participant bidder.
    - Contract balance now is equal to cheaters' deposits + manager's deposit.
    - The contract balance is now divided (integer division) on the honest bidders (including the winner).
    - The participant who called the function will get his share (as the other honest bidders) and the rest of integer division.
    - Note: after the auction has ended the auction state will be set to Ended, which means that the contract is only valid for one auction.

Bonus discussion: after the winner is decided, try to propose a scheme which ensures that the producer will give access to the recording to the winner.

As shown in the auction protocol description: winner can only send money to manager in finalization time and he has no guarantee that the manager will send him the recording as the manager has no incentive: when the manager call endAuctionByManager after the winner sent the money to the manager, the manager gets his deposit value back whether he sent the recording or not.

First intuition:

In order to give incentive to the manager to send the true recording to the winner we can lock his deposit money and in order to get his money back, the winner should confirm that he received the true recording, we can achieve that by modifying the following in the code (modifications are highlighted):

```
bool public _winnerReceivedRecording = false;
modifier winnerReceivedRecording (uint256 time){
    require(_winnerReceivedRecording == true, "winner didn't receive recording.");
    _;
  }
function winnerSendMoneyToManager()
    public
    inFinalizationTime(block.timestamp)
```

```
    calledByWinner()
    checkWinnerTxValue()
    payable
{
    auctionManager.transfer(msg.value + BiddersToDeposits[msg.sender]);
    // winner's deposit now is equal to zero as he transferred his money to manager.
    BiddersToDeposits[msg.sender] = 0;
    _winnerReceivedRecording = true;

    }

function endAuctionByManager()
    public
    inFinalizationTime(block.timestamp)
    calledByAuctionManager()
  winnerReceivedRecording()
  {
    .....
  }
```

I didn't change the function endAuctionByParticipant() as this function exists to help participants get back their deposits in case the manager decided to drop out in the middle of the auction. This function can only be called **after** finalization time and the winner should confirm receiving the recording **in** the finalization time.

However, the winner cannot call this function, neither will receive a share from the cheaters' deposits, the reason is the winner could be dishonest and even though he received the recording he won't call winnerSendMoneyToManager function and accordingly denying the manager his money, and if the endAuctionByParticipant() allows the winner to take back his deposit then he would also be able to refund his deposit and therefore, taking the recording for free (If not gaining more as there could be cheaters and he will take his share from their deposits.).

Analysis:

1st case: Winner and Manager are honest

- In the Finalization phase, the manager will send to the winner the true recording. The winner will call winnerSendMoneyToManager and the manager will receive his money [second highest bid value] from the winner. The honest bidders will refund their deposits.
- The manager call endAuctionByManager and collect his deposit money as mentioned in the protocol description.

  ⇨ No losses as both parties acting honestly.

2nd case: Winner honest, Manager dishonest

- In the finalization phase, the manager will send a false recording to the winner. The winner won't call winnerSendMoneyToManager. The honest bidders will refund their deposits.
- In the finalization phase, the manager can't call endAuctionByManager so his deposit is locked.
- After finalization phase, participants will end the auction.

    ⇨ If the manager does not send the true recording to the winner, **both parties will lose an amount of money = to their initial deposit**. Because the money is locked in the contract until the manager behaves honestly and **he must act honestly within the finalization phase**.
    ⇨ The only way for the manager to refund his deposit is by calling the endAuctionByManager **in finalization phase**, but he cannot call this function as the winner didn't confirm that he received the recording.
    ⇨ There is no way the winner can refund his deposit as function refundDeposit() has notCalledByWinner() condition and endAuctionByParticipants exclude the winner.

3rd case: Winner dishonest, Manager honest

- In the finalization step, the manager will send a true recording to the winner. But the dishonest winner won't call winnerSendMoneyToManager. The honest bidders will refund their deposits.
- The manager cannot call endAuctionByManager.
- After Finalization time, the participants will end the auction.
    ⇨ If the winner does not call winnerSendMoneyToManager () even though he received the true recording, the winner got the true recording with only the deposit money.
    ⇨ The manager has lost the bidding money amount that he should get as a result of sending the whole true recording to winner + he lost the deposit money.

In case 2, the loss for the dishonest manager and the honest winner is equal (initial deposit amount) but in case 3 the loss of the honest manager is much bigger because he sent the whole true recording without getting any revenue + he lost his initial deposit money.

Modifications:

To reduce the loss in the 3rd case we can modify the protocol, instead of the manager giving access to the whole recording to the winner at once, instead he gives the winner access to different parts of the recording on multiple rounds.

Modify the function winnerSendMoneyToManager to receive different money amounts from the winner on different rounds, in the last round it set _winnerReceivedRecording Boolean to true so that the manager can refund his deposit.

Code modifications:

```
Uint public numberOfParts = 0;
Uint public partFee = 0;
Uint public currPart = 0;
Uint public rem = 0;
Function public winnerSetNumberOfParts(uint number_of_parts)
Public
inFinalizationTime(block.timestamp)
calledByWinner()
{
        numberOfParts = number_of_parts;
        uint winner_rem_money = secondHighestBid – deposit;
        partFee = winner_rem_money /number_of_parts;
        mod = winner_rem_money %number_of_parts;
        rem = winner_rem_money  - partFee*mod ;
}

modifier checkWinnerTxValue(){
    if (bidderToDeposit[msg.sender] != 0){ // first time
        require( msg.value == 0, "Winner tx value is not equal to part fee."); }
    Else { // not first time
        require( msg.value == partFee, "Winner tx value is not equal to part fee.");}
    _;
  }
bool public _winnerReceivedRecording = false;
function winnerSendMoneyToManager()
    public
    inFinalizationTime(block.timestamp)
    calledByWinner()
    checkWinnerTxValue()
    payable
  {
        // the first time he calls this function, he sends the deposit value + integer division
rem to the manager
    If (bidderToDeposit[msg.sender] != 0){
        auctionManager.transfer(BiddersToDeposit[msg.sender]+rem);
        BiddersToDeposits[msg.sender]=0;
```

```
     }
// if not first time, then transfer part fee value
    else {auctionManager.transfer(msg.value);
              currPart ++;
      if(currPart == numberOfParts){
           _winnerReceivedRecording = true;
           }
    }

}
```

Winner sets number of parts by calling winnerSetNumberOfParts().

At each round: the manager send the recording part to winner first then the winner send a part of the money to manager by calling winnerSendMoneyToManager().

Then repeat until it is the last round and sets _winnerReceiedRecording to true so that the manager can refund his initial deposit.

Revisit the 3 cases after modifications

1st case: Winner and Manager are honest

- ⇨ Same as before. No losses as both parties acting honestly.

2nd case: Winner honest, Manager dishonest

- ⇨ Same as before. Both parties have their deposit locked until the manager acts honestly and send the recording.

3rd case: Winner dishonest, Manager honest

- ⇨ At any round, if the winner does not call winnerSendMoneyToManager even though he received the true recording, the winner will get a part of the true recording for free that is proportional to a money part.
- ⇨ The manager has lost the initial deposit money because he can't call endAuctionByManager but contrarily he did not send the whole recording for free to the winner he only send a small portion for free.

We could modify the code further as at any round where the winner confirms that he received a part of true recording, the manager could get a portion of his initial deposit.

The manager could also set the minimum number of parts that a winner can set in the constructor so that the winner can't choose the number of parts to be equal to one, because that could lead to the manager sending the whole recording to the winner at the beginning and therefore if the winner is dishonest he could simply not send the money to manager.

Another possible approach is to use ZKproof:

- The winner must first send (the second highest bidding value – deposit) to the contract.
- Then, the manager must provide a zkproof that he encrypted the recording file using the winner's public key in a determined time duration.
- If the manger failed to provide this proof within the determined time duration, then the winner will be able to retrieve all his money back (deposit + rest of the second highest bidding value).
- If the manager succeeds in providing the proof within the determined time duration, the contract will send him the winner's money.

Even though this approach guarantee that the winner receives a file before his bidding money will go to the manager, there is no guarantee that this file contains the true music recording. However the winner is already bidding on an unknown recording so he won't be able to tell a true from a fake recording in most cases unless the manager releases the true recording to the public.