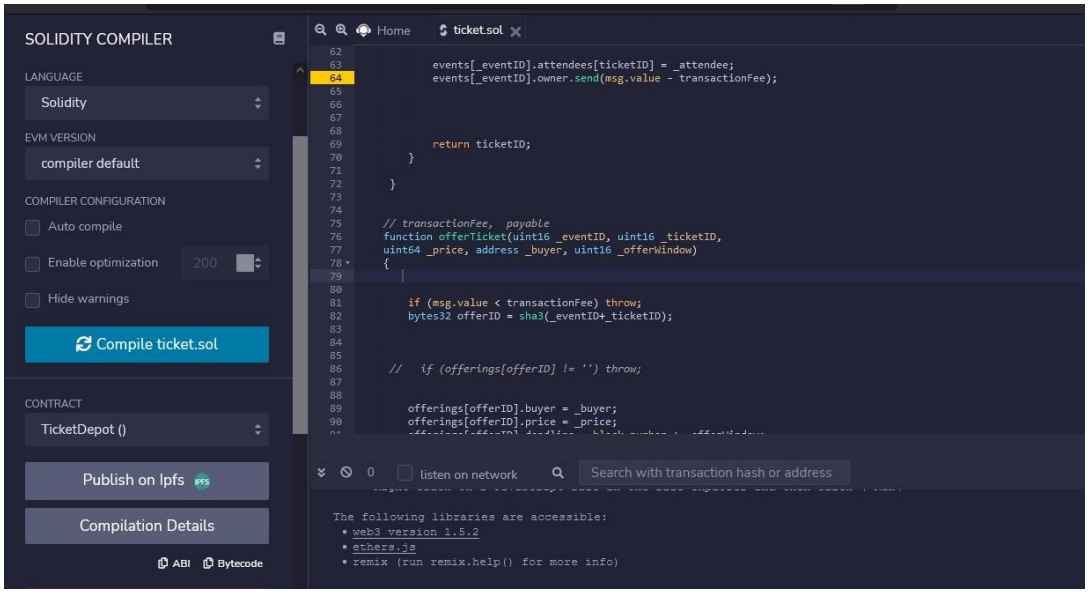
Name: Divya Mahur

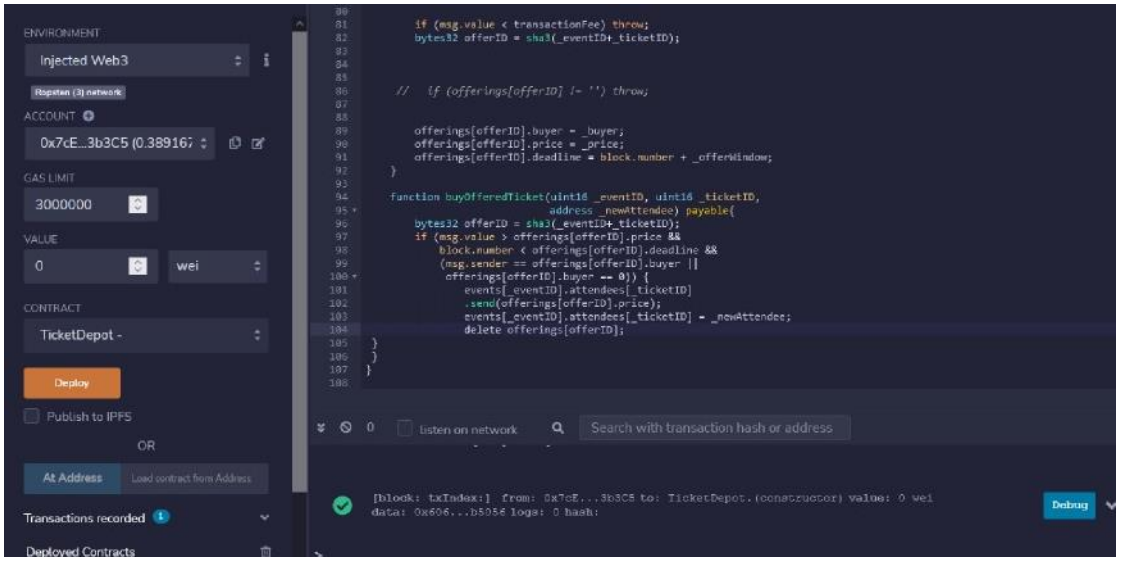
Roll no: 18BCE106

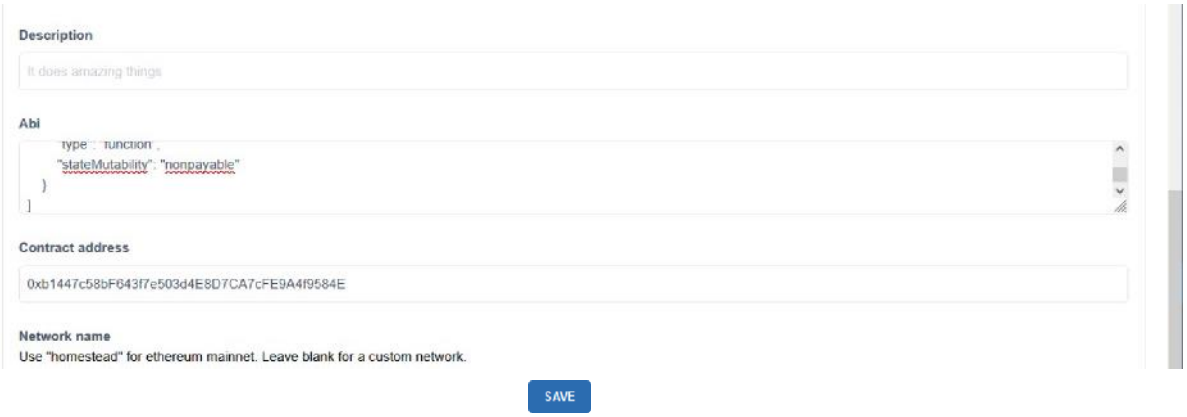
Subject: BCT

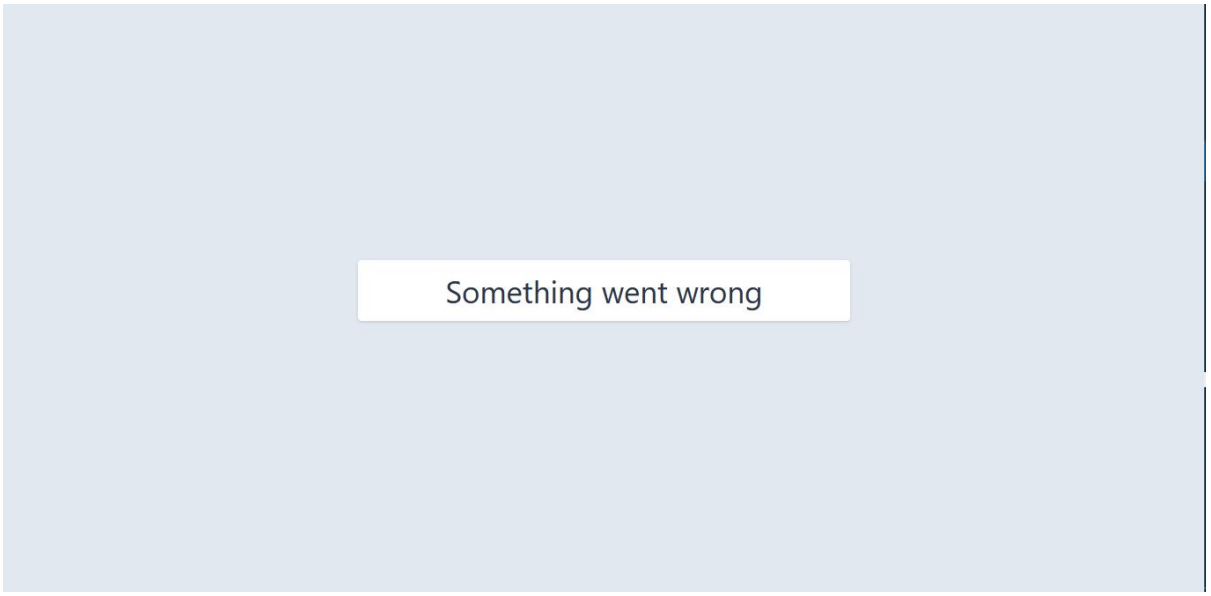
Practical-9

Title: Ticket Solution






Description
It does amazing things
Abi
"type" : "function" ,
"stateMutability" : "nonpayable"
}
]
Contract address
0xb1447c58bF643f7e503d4E8D7CA7cFE9A4f9584E
Network name
Use "homestead" for ethereum mainnet. Leave blank for a custom network.
SAVE


Something went wrong

ABI


```
[
	{
		"constant": false,
		"inputs": [
			{
				"name": "_ticketPrice",
				"type": "uint64"
			},
			{
				"name": "_ticketsAvailable",
				"type": "uint16"
			}
		],
		"name": "createEvent",
		"outputs": [
			{
				"name": "eventID",
				"type": "uint16"
			}
		],
		"payable": false,
		"type": "function",
		"stateMutability": "nonpayable"
	},
	{
		"constant": false,
		"inputs": [
			{
				"name": "_eventID",
```

```
                        "type": "uint16"
                },
                {
                        "name": "_attendee",
                        "type": "address"
                }
        ],
        "name": "buyNewTicket",
        "outputs": [
                {
                        "name": "ticketID",
                        "type": "uint16"
                }
        ],
        "payable": true,
        "type": "function",
        "stateMutability": "payable"
},
{
        "constant": false,
        "inputs": [
                {
                        "name": "_eventID",
                        "type": "uint16"
                },
                {
                        "name": "_ticketID",
                        "type": "uint16"
                },
                {
                        "name": "_price",
                        "type": "uint64"
                },
                {
                        "name": "_buyer",
                        "type": "address"
                },
                {
                        "name": "_offerWindow",
                        "type": "uint16"
                }
        ],
        "name": "offerTicket",
        "outputs": [],
        "payable": false,
        "type": "function",
        "stateMutability": "nonpayable"
},
{
        "constant": false,
```

```
                "inputs": [
                        {
                                "name": "_eventID",
                                "type": "uint16"
                        },
                        {
                                "name": "_ticketID",
                                "type": "uint16"
                        },
                        {
                                "name": "_newAttendee",
                                "type": "address"
                        }

                ],
                "name": "buyOfferedTicket",
                "outputs": [],
                "payable": true,
                "type": "function",
                "stateMutability": "payable"
        },
        {
                "constant": false,
                "inputs": [
                        {
                                "name": "_transactionFee",
                                "type": "uint64"
                        }
                ],
                "name": "ticketDepot",
                "outputs": [],
                "payable": false,
                "type": "function",
                "stateMutability": "nonpayable"
        }
]
```


Contract code

0xb1447c58bF643f7e503d4E8D7CA7cFE9A4f9584E

Code:

```
pragma solidity ^0.4.0;

contract TicketDepot {
   struct Event{
      address owner;
      uint64 ticketPrice;
      uint16 ticketsRemaining;
      mapping(uint16 => address) attendees;
```

```
    }

struct Offering{
    address buyer;
    uint64 price;
    uint256 deadline;
}

uint16 numEvents;
address owner;
uint64 transactionFee;
mapping(uint16 => Event) events;
mapping(bytes32 => Offering) offerings;

function ticketDepot(uint64 _transactionFee){
    transactionFee = _transactionFee;
    owner = tx.origin;

    // sender tx.origin
    // TicketDeport check msg.sender，  onwer
}


// uint16 eventID overflow
// unit64 eventID
function createEvent(uint64 _ticketPrice,
                uint16 _ticketsAvailable) returns (uint16 eventID)
{
    numEvents++;
    // Event
    events[numEvents].owner = tx.origin;
    events[numEvents].ticketPrice = _ticketPrice;
    events[numEvents].ticketsRemaining = _ticketsAvailable;
    return numEvents;
 }
 //

modifier ticketsAvailable(uint16 _eventID){
    _;
    if (events[_eventID].ticketsRemaining <= 0) throw;
}

function buyNewTicket(uint16 _eventID, address _attendee)

//
ticketsAvailable(_eventID) payable returns (uint16 ticketID){

    // msg.value > events[_eventID].ticketPrice + transactionFee


    if (msg.sender == events[_eventID].owner || msg.value >
```

```
        events[_eventID].ticketPrice + transactionFee){
        ticketID = events[_eventID].ticketsRemaining--;

        events[_eventID].attendees[ticketID] = _attendee;
        events[_eventID].owner.send(msg.value - transactionFee);




        return ticketID;
      }

    }


   // transactionFee, payable
   function offerTicket(uint16 _eventID, uint16 _ticketID,
                    uint64_price, address _buyer, uint16 _offerWindow)
   {


      if (msg.value < transactionFee) throw;
      bytes32 offerID = sha3(_eventID+_ticketID);



      if (offerings[offerID] != 0) throw;


      offerings[offerID].buyer = _buyer;
      offerings[offerID].price = _price;
      offerings[offerID].deadline = block.number + _offerWindow;
   }

   function buyOfferedTicket(uint16 _eventID, uint16 _ticketID,
                        address _newAttendee) payable{
      bytes32 offerID = sha3(_eventID+_ticketID);
      if (msg.value > offerings[offerID].price &&
         block.number < offerings[offerID].deadline &&
         (msg.sender == offerings[offerID].buyer ||
         offerings[offerID].buyer == 0)) {
            events[_eventID].attendees[_ticketID]
                  .send(offerings[offerID].price);
            events[_eventID].attendees[_ticketID] = _newAttendee;
            delete offerings[offerID];
 }
 }
}
```