Roll No: 20BCE204
Course: 2CSDE93 - Blockchain Technology
Practical No: 9
Aim: To write a Solidity contract that implements a distributed ticket sales system. Anybody can create an event (specifying the initial price and number of tickets). Anybody can then purchase one of the initial tickets or sell those tickets peer-to-peer. At the event, gate agents will check that each attendee is listed in the final attendees list on the blockchain. (Ethereum programming)

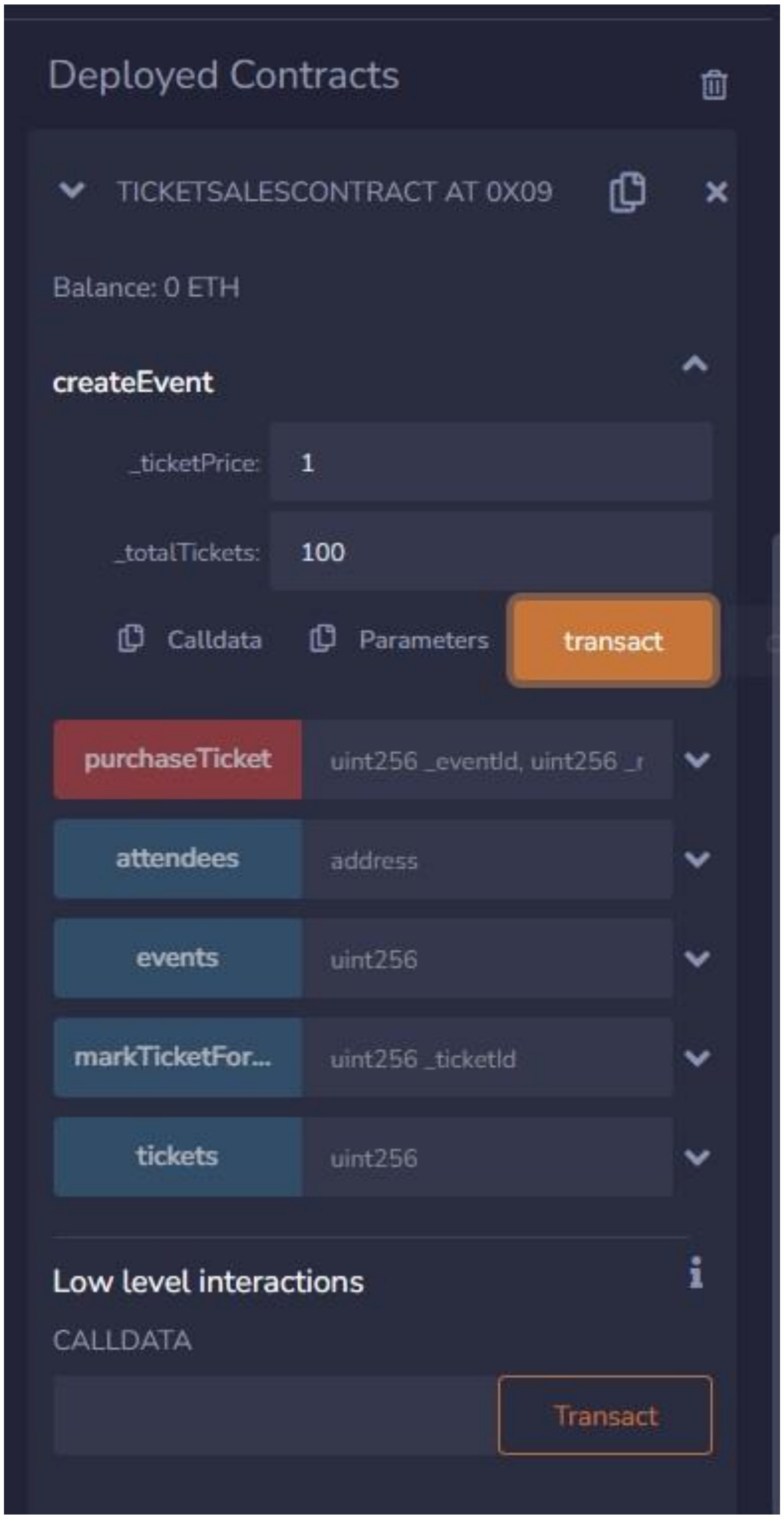
Code:

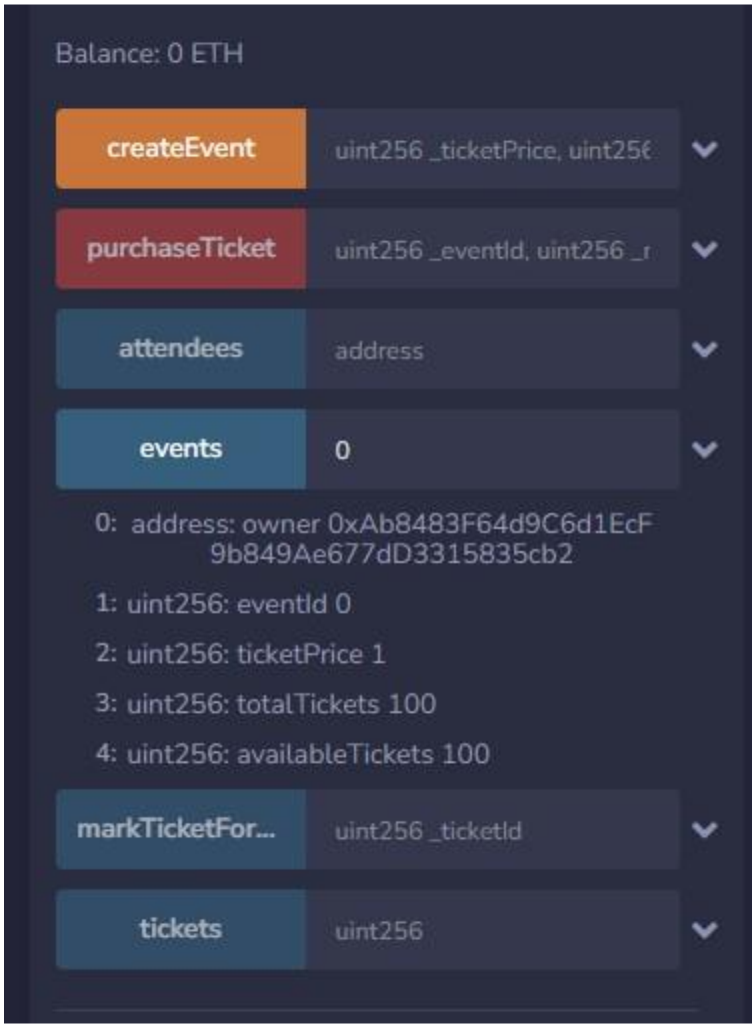
```
// SPDX-License-Identifier: MIT

/* To write a Solidity contract that implements a distributed
ticket sales system. Anybody can create an event (specifying the
initial price and number of tickets). Anybody can then purchase one
of the initial tickets or sell those tickets peer-to-peer. At the event,
gate
agents will check that each attendee is listed in the final attendees list
on the blockchain. (Ethereum programming)
*/
pragma solidity ^0.8.18;

contract TicketSalesContract {
    struct Event {
        address owner;
        uint256 eventId;
        uint256 ticketPrice;
        uint256 totalTickets;
        uint256 availableTickets;
    }
    Event[] public events;

    struct Ticket {
        address holder;
        uint256 ticketId;
        uint256 eventId;
        bool used;
        bool forSale;
    }

    Ticket[] public tickets;
    mapping(address => bool) public attendees;

    event EventCreated(
        uint256 eventId,
        uint256 ticketPrice,
        uint256 totalTickets
    );
    event TicketPurchased(address buyer, uint256 ticketId);
    event TicketSold(address seller, address buyer, uint256 ticketId);

    function createEvent(uint256 _ticketPrice, uint256 _totalTickets)
public {
        uint256 eventId = events.length;
        events.push(
            Event(
                msg.sender,
                eventId,
                _ticketPrice,
                _totalTickets,
                _totalTickets
            )
        );
    }

    function purchaseTicket(
        uint256 _eventId,
        uint256 _noOfTickets
    ) public payable {
        Event storage eventInstance = events[_eventId];
        require(events[_eventId].eventId > 0, "No event found!");

        if (eventInstance.availableTickets < _noOfTickets) {
            // run a  for loop and find out how many other tickets are
marked for the sale.
```

```
            // if the tickets are available than buy them and transfer
money to owner account.
            // else abort the function.

            uint256 isAllAvailable = _noOfTickets;
            for (uint256 j = 0; j < tickets.length; j++) {
                Ticket storage ticket = tickets[j];
                if (
                    isAllAvailable > 0 &&
                    ticket.eventId == _eventId &&
                    !ticket.used &&
                    !ticket.forSale
                ) {
                    isAllAvailable--;
                }
            }

            if (isAllAvailable == 0) {
                require(
                    msg.value > (eventInstance.ticketPrice *
_noOfTickets),
                    "Invalid payment amount"
                );
                for (uint256 j = 0; j < tickets.length; j++) {
                    Ticket storage ticket = tickets[j];
                    if (
                        isAllAvailable > 0 &&
                        ticket.eventId == _eventId &&
                        !ticket.used &&
                        !ticket.forSale
                    ) {
                        ticket.holder = msg.sender;
                        ticket.forSale = false;

payable(eventInstance.owner).transfer(eventInstance.ticketPrice);
                    }
                }
            } else {
                revert("Tickets not available");
            }
```

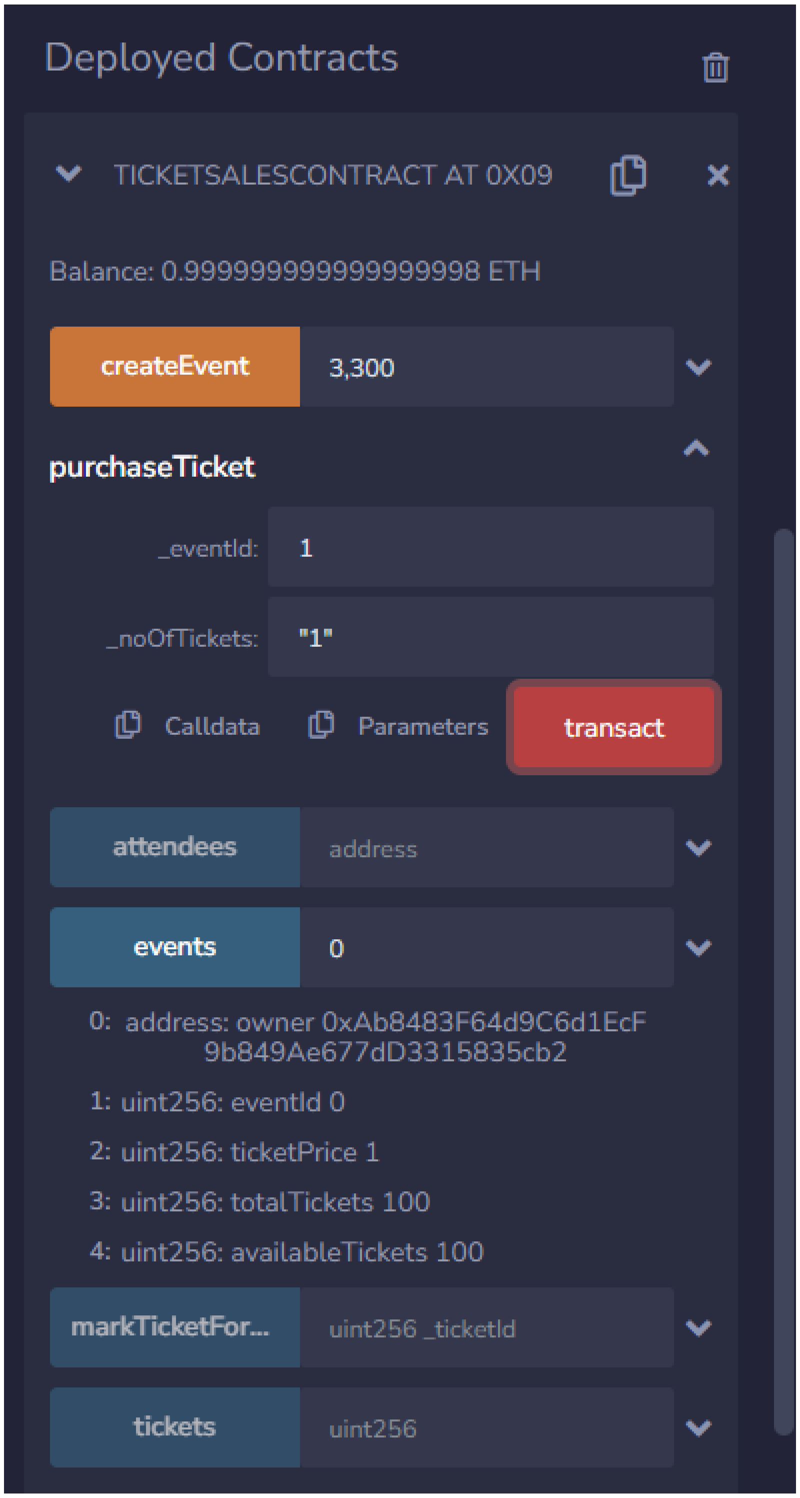
```
        } else {
            require(
                msg.value > (eventInstance.ticketPrice * _noOfTickets),
                "Invalid payment amount"
            );

            Event storage e = events[_eventId];
            e.availableTickets -= _noOfTickets;
            for (uint256 i = 1; i <= _noOfTickets; i++) {
                uint256 ticketId = tickets.length;
                tickets.push(
                    Ticket(msg.sender, ticketId, _eventId, false, false)
                );
            }

payable(eventInstance.owner).transfer(eventInstance.ticketPrice *
_noOfTickets);
        }
    }

    function markTicketForSale(uint256 _ticketId) public view {
        require(tickets[_ticketId].ticketId > 0, "No ticket found!");
        require(
            tickets[_ticketId].holder != msg.sender,
            "The ticket does not belong to you"
        );
        Ticket memory t = tickets[_ticketId];
        t.forSale = true;
    }
}
```

Output:


Deployed Contracts
TICKETSALESCONTRACT AT 0X09
Balance: 0 ETH
createEvent
_ticketPrice: 1
_totalTickets: 100
Calldata
Parameters
transact
purchaseTicket
uint256 _eventId, uint256 _r
attendees
address
events
uint256
markTicketFor...
uint256 _ticketId
tickets
uint256
Low level interactions
CALLDATA
Transact



```
0: address: owner 0xAb8483F64d9C6d1EcF
        9b849Ae677dD3315835cb2
1: uint256: eventId 0
2: uint256: ticketPrice 1
3: uint256: totalTickets 100
4: uint256: availableTickets 100
```



Deployed Contracts

TICKETSALESCONTRACT AT 0X09

Balance: 0.999999999999999998 ETH

createEvent 3,300

purchaseTicket

_eventId: 1

_noOfTickets: "1"

Calldata Parameters transact

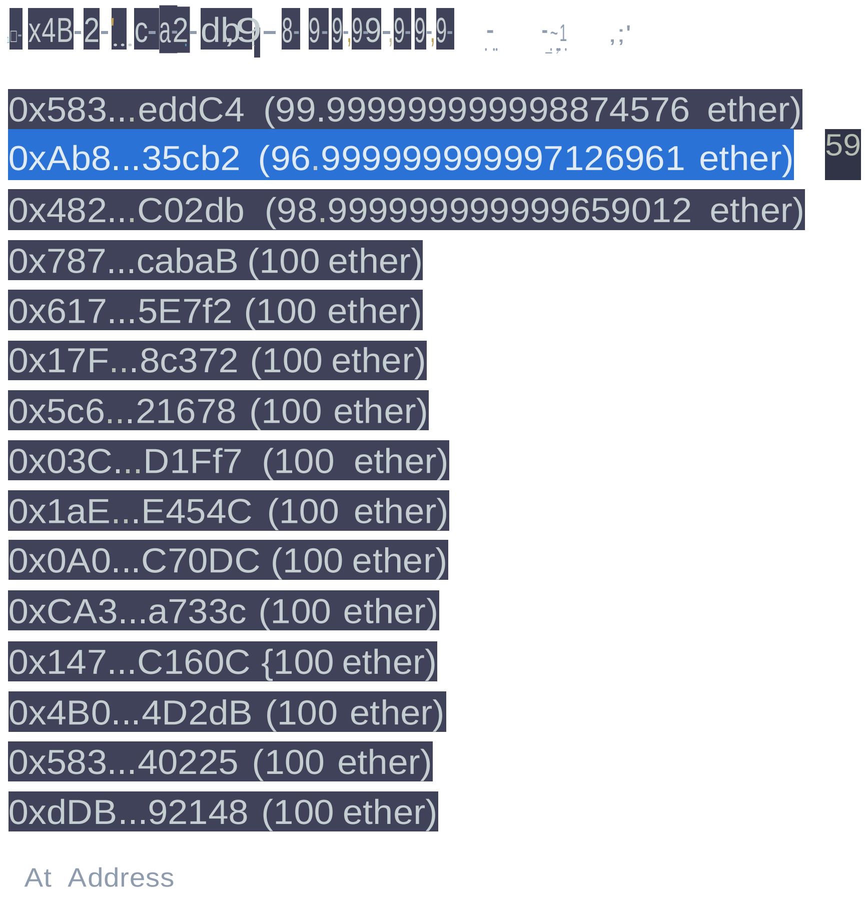
attendees address

events 0

```
0: address: owner 0xAb8483F64d9C6d1EcF
          9b849Ae677dD3315835cb2
1: uint256: eventId 0
2: uint256: ticketPrice 1
3: uint256: totalTickets 100
4: uint256: availableTickets 100
```

markTicketFor... uint256 _ticketId

tickets uint256



```
x4B2...ca2db9 8 9 9 99 9 9 9
0x583...eddC4 (99.999999999998874576 ether)
0xAb8...35cb2 (96.999999999997126961 ether)   59
0x482...C02db (98.999999999999659012 ether)
0x787...cabaB (100 ether)
0x617...5E7f2 (100 ether)
0x17F...8c372 (100 ether)
0x5c6...21678 (100 ether)
0x03C...D1Ff7 (100 ether)
0x1aE...E454C (100 ether)
0x0A0...C70DC (100 ether)
0xCA3...a733c (100 ether)
0x147...C160C {100 ether)
0x4B0...4D2dB (100 ether)
0x583...40225 (100 ether)
0xdDB...92148 (100 ether)
```

At Address