Assignment 4: Ticket sale

Problem statement

Implement a smart contract for selling tickets to events.

- The creator of the contract is the owner.
- The owner specifies the number and price of the tickets.
- People can buy tickets directly from the contract.
- Anyone can validate whether a given address owns a ticket.
- One person can only buy one ticket.
- People can safely swap tickets through the contract.

Example scenario

Sziget Festival decides to use this contract to sell day tickets. They create an instance for 100,000 tickets, setting the price at the wei equivalent of 10,000.

In this case, the ticket identifiers go from 1 to 100,000. People can check the type of any ticket based on this id on the festival's website, e.g. tickets 1 to 10,000 are for Monday, 10,001 to 20,000 are for Tuesday, etc.

Alice and Bob both buy tickets, `#784` and `#10,322` respectively, paying the wei equivalent of 10,000 each. However, Alice wants to go on Tuesday instead, and Bob would rather go on Monday. Thus, they decide to swap. For this, Alice submits a swap offer between their Ethereum addresses. Bob accepts the offer. In the end, Alice owns ticket `#10,322`, while Bob owns ticket `#784`.

On the day of the festival, the festival staff scans Alice's QR code at the entrance. The system checks whether Alice has a ticket and whether that ticket is valid at that time. After this, Alice can go in and enjoy the festival.

Contract interface

- constructor: `constructor(uint numTickets, uint price)`
  - Create a ticket sale with `numTickets` tickets, each sold for `price` wei.
  - The sender of the transaction (`msg.sender`) is the manager.
- buy ticket: `function buyTicket(uint ticketId) payable`
  - The sender of the transaction (`msg.sender`) buys `ticketId` for the specified price.

    - Succeeds only if `ticketId` is a valid ticket identifier, the ticket has not yet been sold, the sender has not bought a ticket yet, and he/she sends the correct amount of ether.
- get ticket id: `unction getTicketOf(address person) public view returns (uint)`
    - Returns the ticket id associated with the address `person`, or `0` in case `person` does not have a ticket.
- offer swap: `function offerSwap(address partner)`
    - The sender of the transaction (`msg.sender`) submits an offer for swapping their ticket with `partner`.
- accept swap offer: `function acceptSwap(address partner)`
    - The sender of the transaction (the buyer) accepts the swap offer of `partner`.
    - Succeeds only if both parties (`msg.sender` and `partner`) have tickets, and there is a swap offer submitted by `partner` for `msg.sender`.
    - After the transaction, the tickets of the two parties are swapped, and the offer is destroyed.

Task 1: Contract skeleton (70 pts)

The contract skeleton is given bellow. **Follow the skeleton, and write codes corresponding to each functions. Attach your .sol file in the submission.**

```
pragma solidity ^0.8.21;

contract TicketSale {

    // <contract_variables>

    // </contract_variables>

    constructor(uint numTickets, uint price) public {
        // TODO
    }

    function buyTicket(uint ticketId) public payable {
        // TODO
    }

    unction getTicketOf(address person) public view returns (uint) {
        // TODO
    }

    function offerSwap(address partner) public {
        // TODO
    }

    function acceptSwap(address partner) public {
        // TODO
    }
```

}

Task 2: Contract skeleton (20 pts)

Compile and Generate Bytecode using 'node compile.js' command. Copy the output from the terminal showing ABI and the Bytecode, paste those in two separate text files and submit those two files.

Task 3: Generate Test Cases (30 pts)

Generate test cases for each of the four methods. Test in Ganache test platform and attch the screenshot showing that your test case passes all the four functions. Submit your test file.

Task 4: Deploy in Goerli test Network (20 pts)

Create a project in infura site for Georli test network. Run 'deply.js' and get the address where your contract is deployed. Visit https://goerli.etherscan.io/ site and verify that your project is uploaded there.

Submit the address where your contract is deployed. Attach the screenshot of bytecode that you see in https://goerli.etherscan.io/ site (don't take screenshot of the bytecode that you see in terminal, it should be from the website)

Task 5: Upload your project in github (10 pts)

Follow this tutorial and upload your project in GitHub **(make sure to delete your secret phrase from the deploy.js file before uploading your project into github).**

https://docs.github.com/en/repositories/working-with-files/managing-files/adding-a-file-to-a-repository

Send your GitHub link in the submission.