

Roll No: 20BCE204

Course Code and Course Name: 2CSDE93 Blockchain Technology

Practical No. 5

Aim: To perform thorough study and installation of Remix IDE and Truffle IDE for deploying Smart Contracts and Decentralized Applications (dapps) and create and deploy a Smart Contract for any application such as finance, healthcare etc.

Code:

```
//SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.18;

// Define a contract for the self-driving car
contract SelfDrivingCar {
    address public owner;
    address public carAddress;
    uint public carSpeed;
    uint public carBalance;
    bool public isDriving;
    mapping(address => uint) public passengerBalances;

    event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
    event CarSpeedUpdated(uint newSpeed);
    event CarBalanceUpdated(uint newBalance);
    event DrivingStatusUpdated(bool isDriving);
    event PassengerBalanceUpdated(address passenger, uint newBalance);

    constructor() {
        owner = msg.sender;
        carAddress = address(this);
        carSpeed = 0;
        carBalance = 0;
        isDriving = false;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can perform this action");
        _;
    }

    modifier notDriving() {
        require(!isDriving, "The car is currently in motion");
```

```

    _;
}

receive() external payable {
    // Handle incoming Ether (e.g., refilling car balance)
    carBalance += msg.value;
    emit CarBalanceUpdated(carBalance);
}

fallback() external {
    // Handle unexpected transactions
}

function transferOwnership(address newOwner) public onlyOwner {
    require(newOwner != address(0), "Invalid address");
    emit OwnershipTransferred(owner, newOwner);
    owner = newOwner;
}

function updateCarSpeed(uint newSpeed) public onlyOwner notDriving {
    carSpeed = newSpeed;
    emit CarSpeedUpdated(newSpeed);
}

function startDriving() public onlyOwner notDriving {
    isDriving = true;
    emit DrivingStatusUpdated(true);
}

function stopDriving() public onlyOwner {
    isDriving = false;
    emit DrivingStatusUpdated(false);
}

function addPassenger(address passenger, uint balance) public onlyOwner {
    require(passenger != address(0), "Invalid address");
    passengerBalances[passenger] = balance;
    emit PassengerBalanceUpdated(passenger, balance);
}

```

```

function payPassenger(address passenger, uint amount) public onlyOwner {
    require(passengerBalances[passenger] >= amount, "Insufficient funds for the passenger");
    passengerBalances[passenger] -= amount;
    carBalance -= amount;
    emit PassengerBalanceUpdated(passenger, passengerBalances[passenger]);
}
}

```

Output:

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows a deployed contract named 'SELFDRIVINGCAR AT 0XF44...0DEB9 (BLC)'. Below the contract name, there are several interactive buttons: 'addPassenger', 'payPassenger', 'startDriving', 'stopDriving', 'transferOwner...', 'updateCarSpe...', 'carAddress', 'carBalance', 'carSpeed', 'isDriving', 'owner', and 'passengerBata...'. The 'Low level interactions' section at the bottom left shows 'CALLDATA'.

The main editor displays the Solidity code for the 'SelfDrivingCar' contract. The code includes events for 'OwnershipTransferred', 'CarSpeedUpdated', 'CarBalanceUpdated', 'DrivingStatusUpdated', and 'PassengerBalanceUpdated'. It also features a 'constructor()' and a 'modifier onlyOwner()'.

The bottom panel shows the transaction log. A transaction to 'SelfDrivingCar.(receive)' is pending. The log entry shows the transaction was successful, with a status of 'true'. The transaction hash is '0x226da296eeef8b01db879b7a45293fe984f31541ff04a9e2343362157c118a7'. The block hash is '0x8bd8868adb5649a842dcb5284d3c968437a6983b424dc91f1284988bc080b6'. The block number is '8'. The transaction cost is '24760 gas' and the execution cost is '3788 gas'.