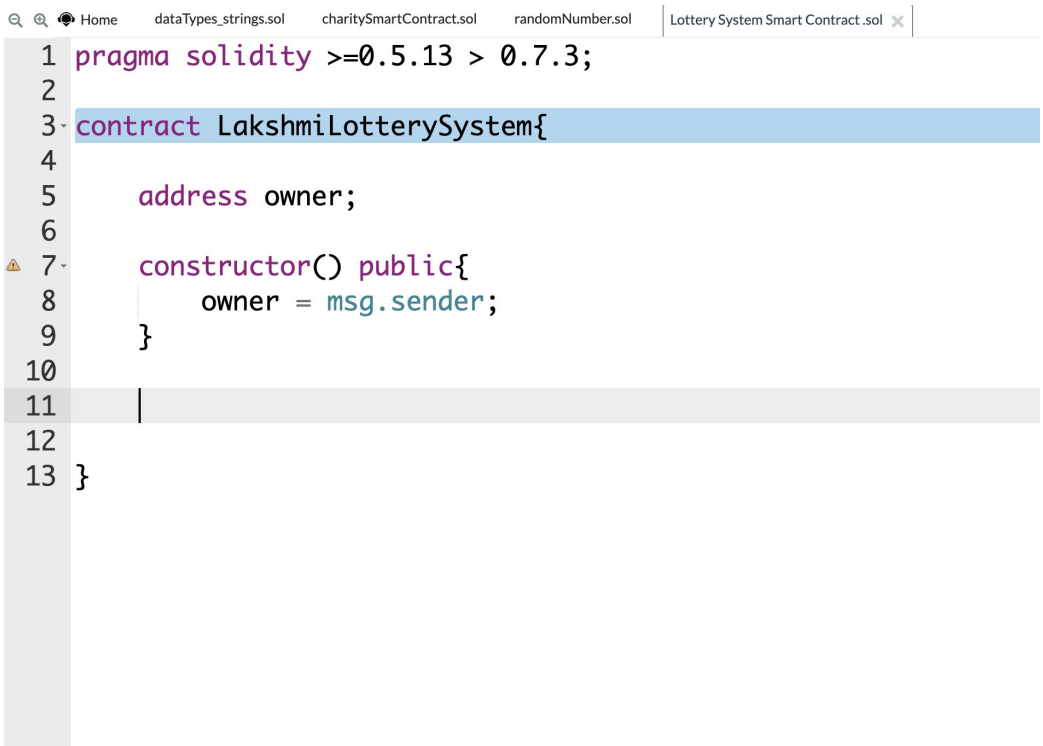
Project #2 - LetsUpgrade Lakshmi Lottery Ethereum

Step 1

What is Lottery System ?

1. We need to develop a smart contract
2. Users should buy lottery tickets using ethereum - Ticket Value >= 1 Eth
3. The Deploying EOA is going to be Owner
4. The owner will have access to pick a Random Winner,
5. The owner will have the access to pause and destroy the smart Contract
6. There should be reset of the smart contract once the Winner is picked for the next round // HOME WORK
7. Only one person can buy one ticket

Step 2


Home
dataTypes_strings.sol
charitySmartContract.sol
randomNumber.sol
Lottery System Smart Contract .sol
1 pragma solidity >=0.5.13 > 0.7.3;
2
3 contract LakshmiLotterySystem{
4
5 address owner;
6
7 constructor() public{
8 owner = msg.sender;
9 }
10
11
12
13 }

Step 3

Home | dataTypes_strings.sol | charitySmartContract.sol | randomNumber.sol | Lottery System Smart Contract .sol | dataType-Address.sol | dataTypes_Maps.sol | 7 tabs

```
1  pragma solidity >=0.5.13 < 0.7.3;
2
3  contract LakshmiLotterySystem{
4
5      address owner;
6
7      mapping(address => uint) public addressOfLotteryParticipants;
8
9      constructor() {
10         owner = msg.sender;
11     }
12
13     function reciveEtherForParticpation() payable public{
14         require(msg.value>= 1 ether, " You require minumum 1 ether to participate in this lottery");
15         addressOfLotteryParticipants[msg.sender] = msg.value;
16     }
17
18     
19
20
21
22 }
```

Step 4

Home | Lottery System Smart Contract .sol

```
1  pragma solidity >=0.5.13 < 0.7.3;
2
3  contract LakshmiLotterySystem{
4
5      address owner;
6
7      mapping(address => uint) public addressOfLotteryParticipants;
8      address[] addressOfPartipant;
9
10     constructor() {
11         owner = msg.sender;
12     }
13
14     function reciveEtherForParticpation() payable public{
15         require(msg.value>= 1 ether, " You require minumum 1 ether to participate in this lo
16         addressOfLotteryParticipants[msg.sender] = msg.value;
17         addressOfPartipant.push(msg.sender);
18     }
19
20
21     function randomNumberFunction() public view returns(uint){
22         uint randomNumber = uint(keccak256(abi.encodePacked(block.difficulty,
23         block.timestamp, msg.sender))) % 100;
24         return(randomNumber);
25     }
26
27
```

Step 5

Home | Lottery System Smart Contract .sol

```
14    function reciveEtherForParticpation() payable public{
15        require(msg.value>= 1 ether, " You require minumum 1 ether to participate in t
16        addressOfLotteryParticipants[msg.sender] = msg.value;
17        addressOfPartipant.push(msg.sender);
18    }
19
20
21    function randomNumberFunction() private onlyOwner returns(uint){
22        uint randomNumber = uint(keccak256(abi.encodePacked(block.difficulty,
23        block.timestamp, msg.sender,"Prabhpreet", addressOfPartipant))) % addressOfPar
24        return(randomNumber);
25    }
26
27
28    modifier onlyOwner(){
29        require(msg.sender == owner, "Owner only has access to this");
30        _;
31    }
32
33
34
35
36
37
38
39
```

ContractDefinition Lakshm

Step 6

```
20
21      function randomNumberFunction() private onlyOwner returns(uint){
22          uint randomNumber = uint(keccak256(abi.encodePacked(block.difficulty,
23          block.timestamp, msg.sender,"Prabhpreet", addressOfPartipant))) % addressOfPartip
24          return(randomNumber);
25      }
26
27      function transferEhterToWinner() public onlyOwner{
28          uint randomWinner = randomNumberFunction();
29          address payable winner = payable(addressOfPartipant[randomWinner]);
30          winner.transfer(address(this).balance);
31
32      }
33
34
35      modifier onlyOwner(){
36          require(msg.sender == owner, "Owner only has access to this");
37          _;
38      }
39
40
41
```

Step 7

```
    -

    function reciveEtherForParticpation() payable public{
        require(msg.value>= 1 ether, " You require minumum 1 ether to participate in this lottery");
        require(contains(msg.sender)==0, " You are already a part of the lottery");
        addressOfLotteryParticipants[msg.sender] = msg.value;
        addressOfPartipant.push(msg.sender);
    }

    function randomNumberFunction() private onlyOwner returns(uint){
        uint randomNumber = uint(keccak256(abi.encodePacked(block.difficulty,
        block.timestamp, msg.sender,"Prabhpreet", addressOfPartipant))) % addressOfPartipant.length;
        return(randomNumber);
    }

    function transferEhterToWinner() public onlyOwner{
        uint randomWinner = randomNumberFunction();
        address payable winner = payable(addressOfPartipant[randomWinner]);
        winner.transfer(address(this).balance);
    }

    modifier onlyOwner(){
        require(msg.sender == owner, "Owner only has access to this");
        _;
    }

    function contains( address _addr)  private onlyOwner returns(uint) {
        return addressOfLotteryParticipants[_addr];
    }
}
```

Step 8

Deploy the Whole in Live Environment - Rickby