Experiment No: 09

● **Aim: To implement a lottery smart contract in Solidity.**

● **Theory:**

Solidity – Types

Solidity is a statically typed language, which implies that the type of each of the variables should be specified. Data types allow the compiler to check the correct usage of the variables. The declared types have some default values called Zero-State, for example for bool the default value is False. Likewise other statically typed languages Solidity has Value types and Reference types which are defined below:

Value Types

Value-type variables store their own data. These are the basic data types provided by solidity. These types of variables are always passed by value. The variables are copied wherever they are used in function arguments or assignments. Value type data types in solidity are listed below:

1. **Boolean**: This data type accepts only two values True or False.
2. **Integer**: This data type is used to store integer values, int, and uint are used to declare signed and unsigned integers respectively.
3. **Fixed Point Numbers**: These data types are not fully supported in solidity yet, as per the Solidity documentation. They can be declared as fixed and unfixed for signed and unsigned fixed-point numbers of varying sizes respectively.
4. **Address**: Address hold a 20-byte value which represents the size of an Ethereum address. An address can be used to get a balance or to transfer a balance by balance and transfer method respectively.
5. **Bytes**: Although bytes are similar to strings, there are some differences between them. bytes used to store a fixed-sized character set while the string is used to store the character set equal to or more than a byte. The length of bytes is from 1 to 32, while the string has a dynamic length. Byte has the advantage that it uses less gas, so better to use when we know the length of data.
6. **Enums**: It is used to create user-defined data types, used to assign a name to an integral constant which makes the contract more readable, maintainable, and less prone to errors. Options of enums can be represented by unsigned integer values starting from 0.

To develop the Smart Contract for simple lottery system following are the steps

1) **Licenses identifier**

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

2) **Version of Solidity :**

```
3   pragma solidity ^0.8.21;
```

3) **Global Variable**

- **Owner**
- **Players**

```
contract Lottery {
    address public owner;
    address payable[] public players;

    constructor() {
        owner = msg.sender;
    }
```

4) **Functions: Enter Lottery**

```
function getBalance() public view returns (uint) {
    return address(this).balance;
}

function getPlayers() public view returns (address payable[] memory) {
    return players;
}

function enter() public payable {
    require(msg.value > .01 ether);

    // address of player entering lottery
    players.push(payable(msg.sender));
}
```

- **Get Random Numbers**

```
function getRandomNumber() public view returns (uint) {
    return uint(keccak256(abi.encodePacked(owner, block.timestamp)));
}
```

- **Pick the Winner –**

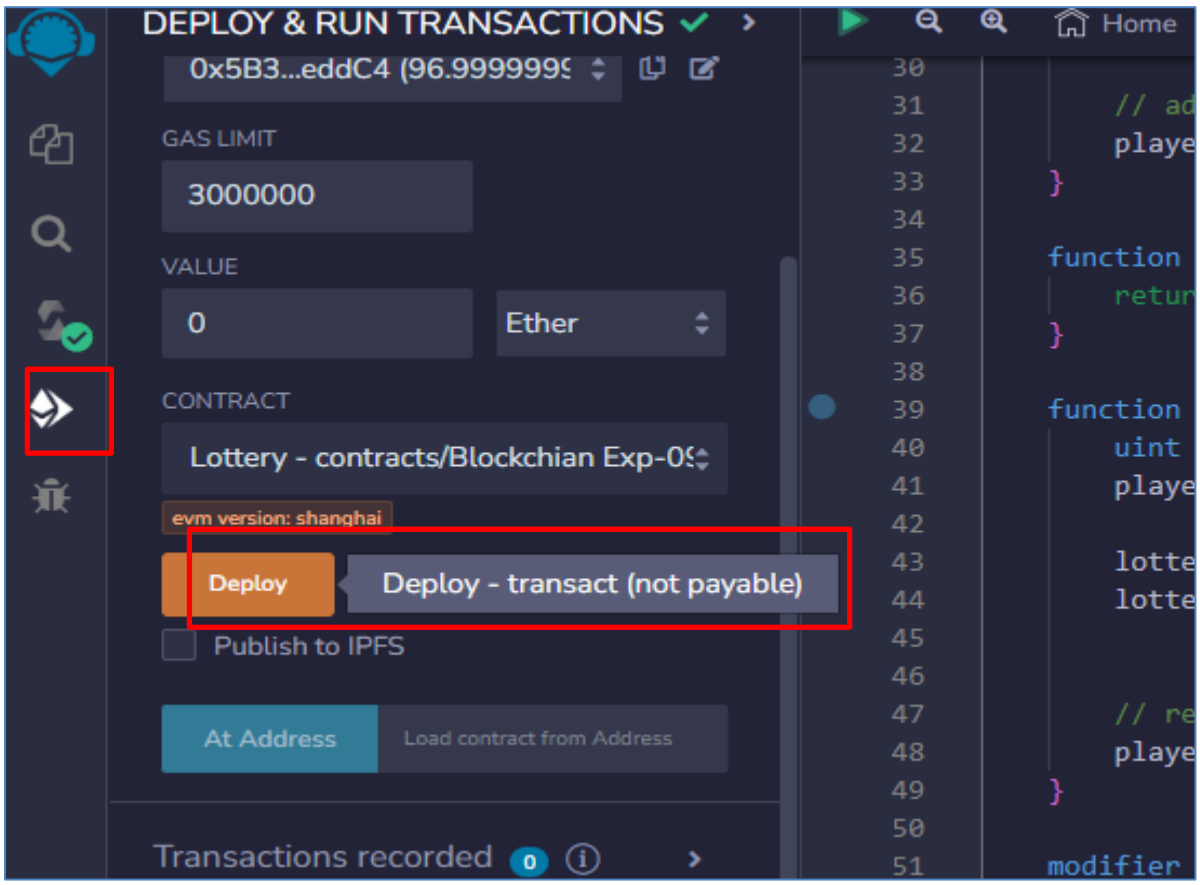
```
function pickWinner() public onlyOwner {
    uint index = getRandomNumber() % players.length;
    players[index].transfer(address(this).balance);

    // reset the state of the contract
    players = new address payable[](0);
}
```

5) **Set the Owner as only Modifier**

```
51      modifier onlyowner() {
52          require(msg.sender == owner);
53          _;
54      }
55  }
```

6) Compile the Code

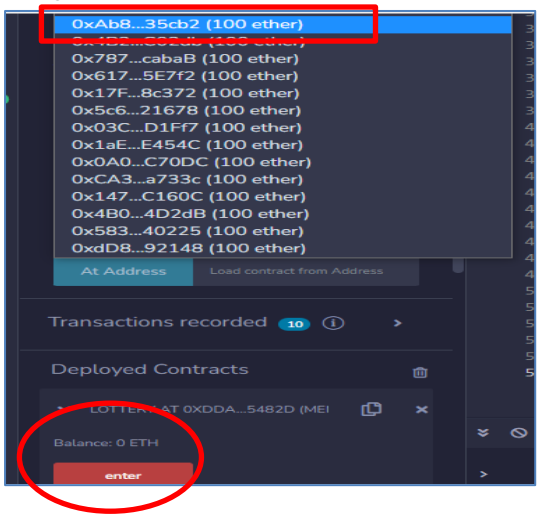
7) Deploy the Code:



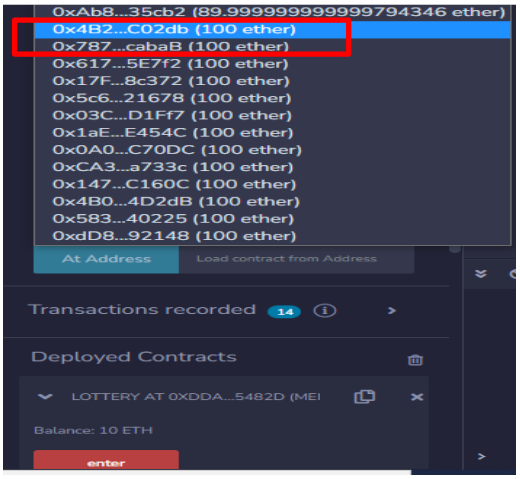
8) Enter the players

Note : 1 ETH = 1000000000000000000 Wei

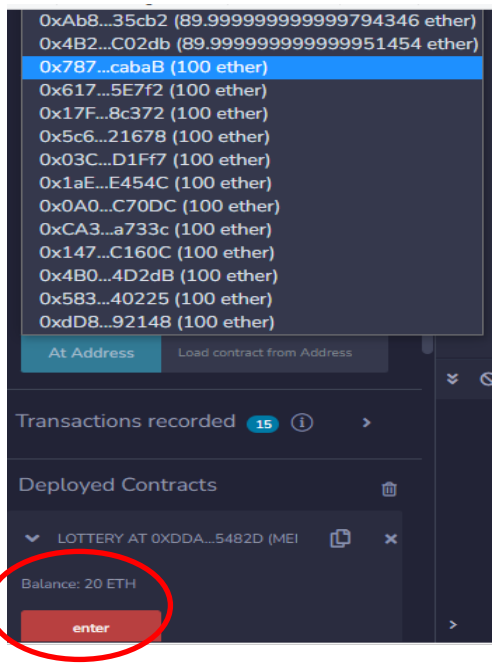
Player -1



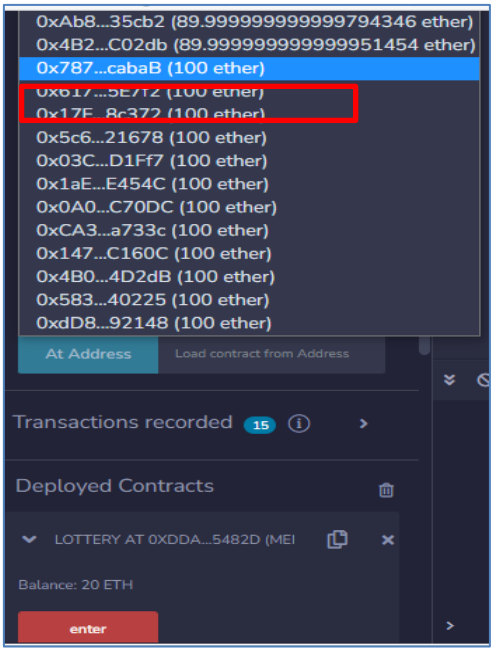
player -2



Player -3

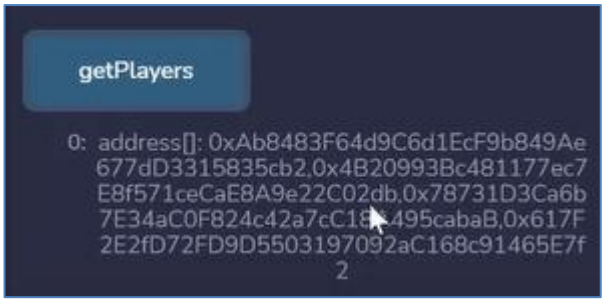


Player -4



9) **Check the Owner :** 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

10) Check players-

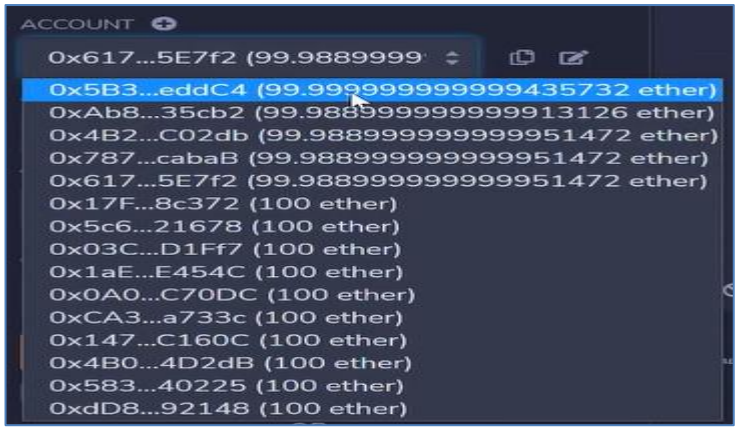


11) Check the balance it should be sum of enter player amount

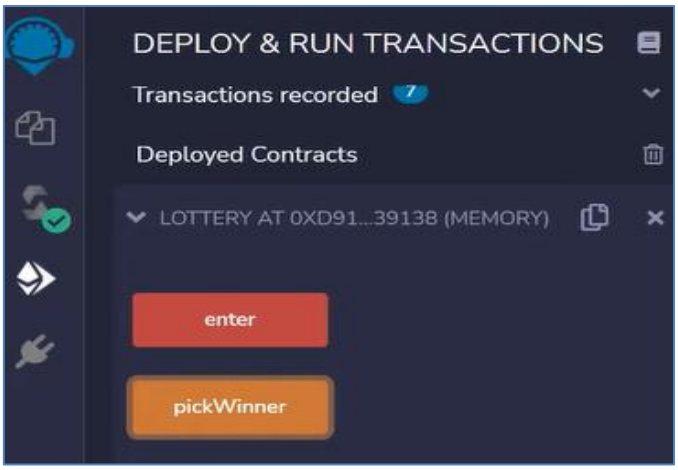
12) Pick the winner:

a. Please note only owner Account can pick the winner

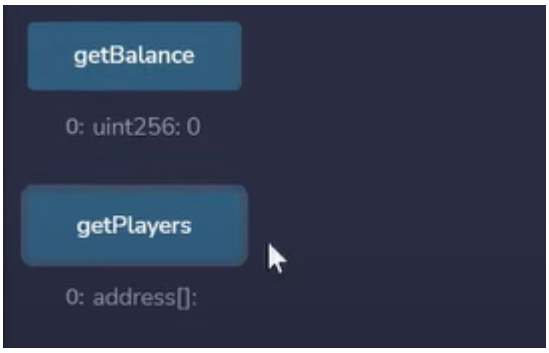
b. Please check the Balance of each player before picking the winner



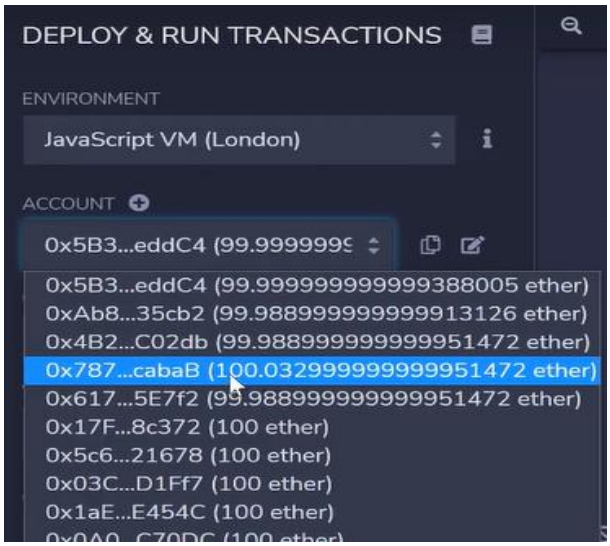
c. After Picking up the winner – All amount should be reflect in the winner Account



d. Now balance should be zero as winner has been picked now & there should not be any player too



e. Check the winner Account balance



Program:

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

pragma solidity ^0.8.11;

contract Lottery {
    address public owner;
    address payable[] public players;

    constructor() {
        owner = msg.sender;
    }

    function getBalance() public view returns (uint) {
        return address(this).balance;
    }

    function getPlayers() public view returns (address payable[] memory) {
        return players;
    }

    function enter() public payable {
        require(msg.value > .01 ether);
```

```
        // address of player entering lottery
        players.push(payable(msg.sender));
    }

    function getRandomNumber() public view returns (uint) {
        return uint(keccak256(abi.encodePacked(owner, block.timestamp)));
    }
```

```
    function pickWinner() public onlyOwner {
        uint index = getRandomNumber() % players.length;
        players[index].transfer(address(this).balance);

        // reset the state of the contract
        players = new address payable[](0);
    }

    modifer onlyOwner() {
      require(msg.sender == owner);
      _;
    }
}
```

- **Conclusion:**

Hence, we have successfully implemented lottery smart contract in solidity