Experiment No: 10

●**Aim: To implement Mini Project using Smart Contract - solidity. Small bank with following functionality –**

- **Deposit**
- **Withdraw**
- **View Balance.**

●**Theory:**

1. Following is the code for Small Bank implementation which having above functionality i.e Deposit, Withdraw, View Balance.

This smart contract includes the following functionalities:

Deposit (deposit function): Users can deposit Ether into their account.

Withdraw (withdraw function): Users can withdraw a specified amount from their account, provided they have a sufficient balance.

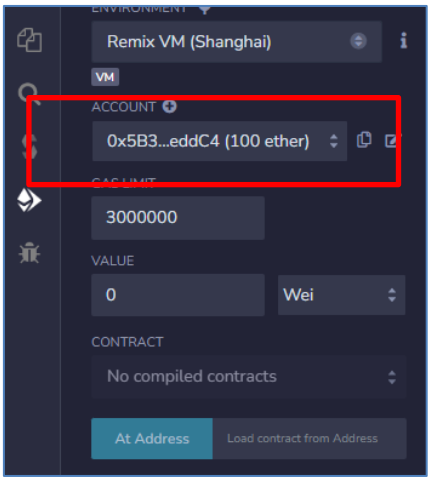
View Balance (viewBalance function): Users can check their account balance.

Contract Balance (contractBalance function): Only the contract owner (the address that deployed the contract) can check the contract's balance. This is often used to check the total balance of the bank.

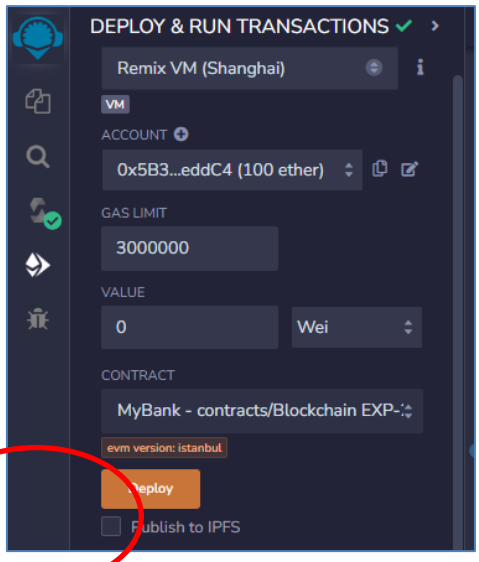
```
1)
2) // SPDX-License-Identifier: Unlicensed
3) pragma solidity ^0.6.12;
4)
5) contract MyBank
6) {
7)     mapping(address=> uint ) private _balances;
8)     address public owner;
9)     event LogDepositeMade(address accountHoder, uint amount );
10)
11)    constructor () public
12)     {
13)         owner=msg.sender;
14)         emit LogDepositeMade(msg.sender, 1000);
15)     }
16)
17)
18)
19)
20)
21)
```

```
22)
23)     function deposite() public payable  returns (uint)
24)        {
25)
26)        require ((_balances[msg.sender] + msg.value)
   >  _balances[msg.sender] && msg.sender!=address(0));
27)        _balances[msg.sender] += msg.value;
28)        emit LogDepositeMade(msg.sender , msg.value);
29)        return _balances[msg.sender];
30)        }
31)
32)        function withdraw (uint withdrawAmount) public  returns (uint)
33)        {
34)
35)                require (_balances[msg.sender] >= withdrawAmount);
36)                require(msg.sender!=address(0));
37)                require (_balances[msg.sender] > 0);
38)                _balances[msg.sender]-= withdrawAmount;
39)                msg.sender.transfer(withdrawAmount);
40)                emit LogDepositeMade(msg.sender , withdrawAmount);
41)                return _balances[msg.sender];
42)
43)        }
44)
45)        function viewBalance() public view returns (uint)
46)        {
47)            return _balances[msg.sender];
48)        }
49)}
```

2. Compile & Deployment :
   1. Contract Owner who has deployed the contract : Owner also can be sender 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

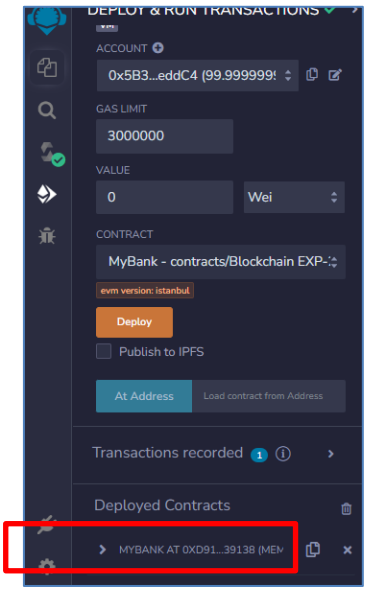


   2. Sender (Sender will not be Owner unless we are announce the owner ship) - 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
   3. Deploy the Code using following Step :



4. Owner of account & Contract Address are showing in Debug Logs as below

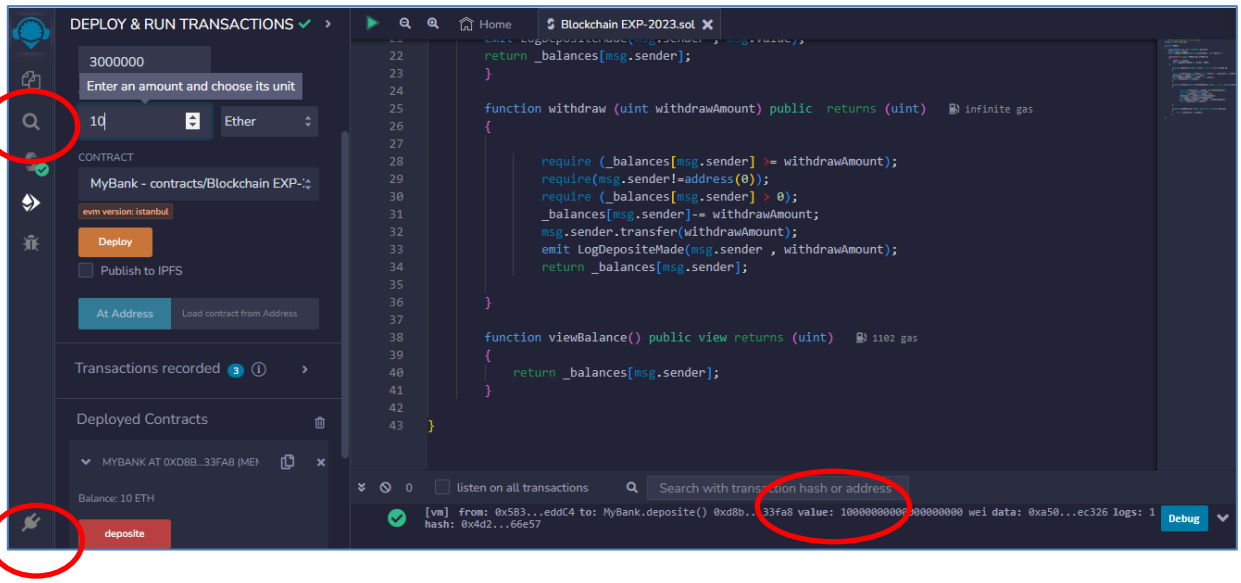
- Contract address (from where code deployed): 0xd9145CCE52D386f254917e481eB44e9943F39138
- Owner : 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4



3. Testing of Project

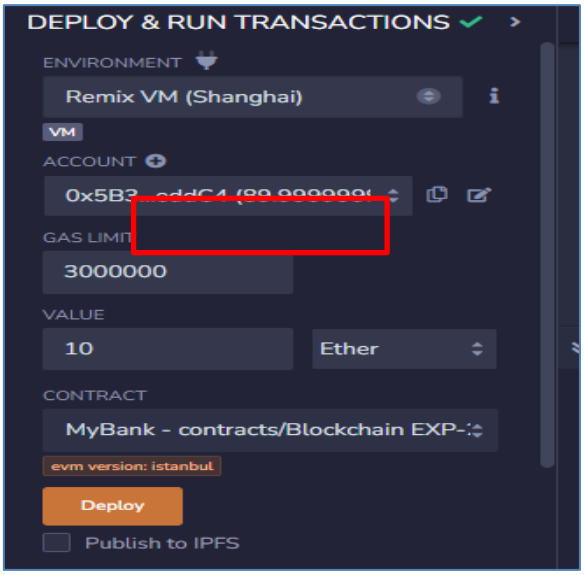
1. Deposit Function :

- Please enter the Eth which you need to deposit- in our case “10” & Click on” Deposit” in deployed contracts section
  Please note 1 ETH = 10000000000000000000 Wie



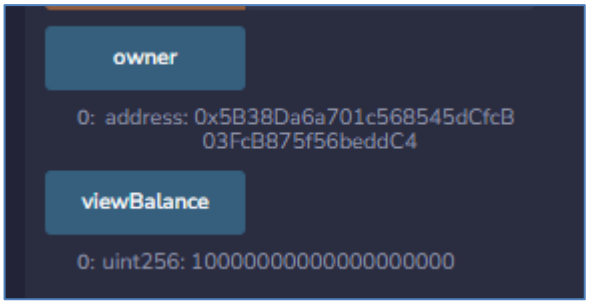
- Logs are blow showing deposit of 10 Eth with Account holder address

- Msg.sender has deducted by 10 Eth and showing in Owner Account

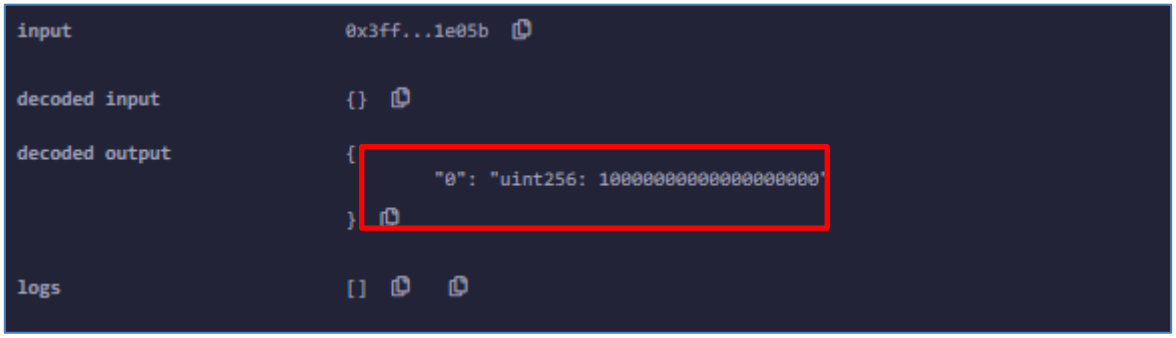


2. To check Balance Function :

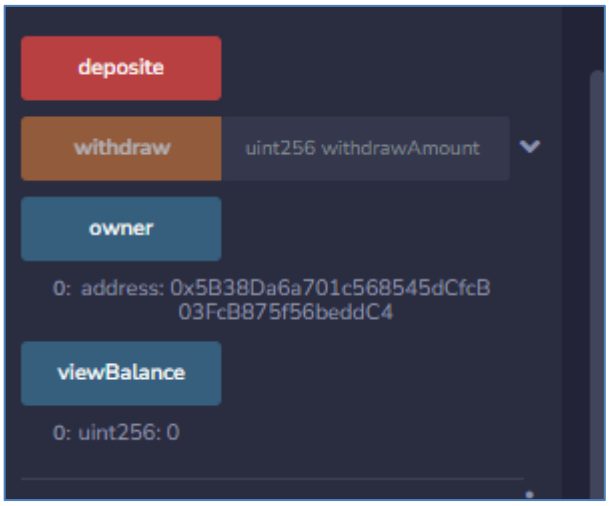
- Click on View Balance : Showing 10 ETH as we have deposited in above step



Logs :

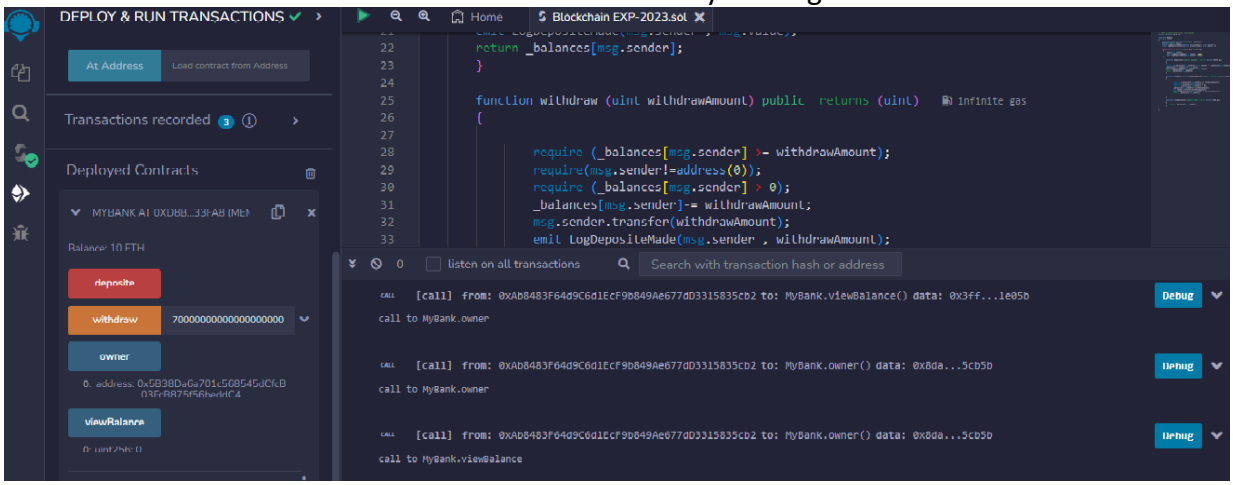


- There is no balance in sender account as we don't deposite any amount but owner will be same

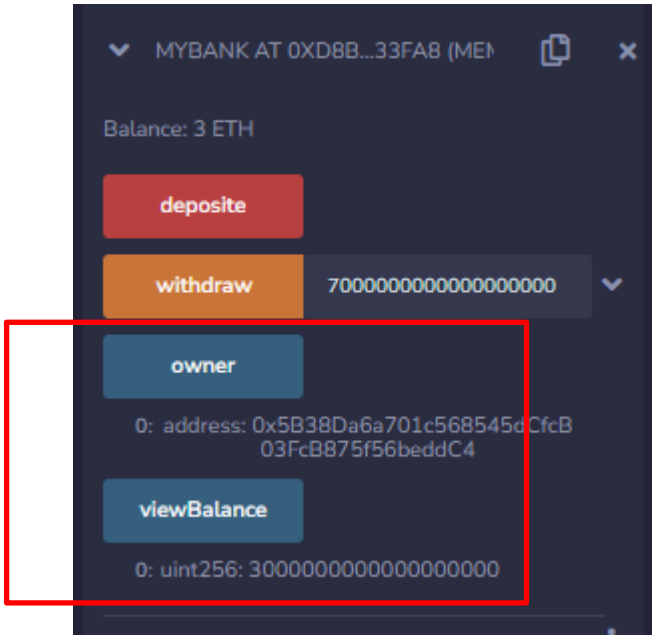


3. Withdraw Function :

- We will withdraw the 7ETH from owner Account by clicking withdraw



-Owner Account showing 3ETH as we have withdrawn the 7 ETH



-Logs are below

- **Conclusion:** Hence, we have test small bank functionality in solidity successfully.