**PRACTICAL 9**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name:** | Harsh Shah | **Roll No.:** | 21BCP359 |
| **Division:** | 6 | **Batch:** | G11 |
| **Aim:** | Deploying Banking Application through Meta Mask. | | |

**Program**

// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.8.2 <0.9.0;

contract MyBank {

mapping(address => uint256) balances;

address public owner;

event Deposit(address indexed account, uint256 amt);

event Withdraw(address indexed account, uint256 amt);

event Transfer(address indexed from, address indexed to, uint256 amt);

modifier onlyOwner() {

require(msg.sender == owner, "Only owner can perform action");

\_;

}

constructor() {

owner = msg.sender;

}

function deposit() public payable {

require(msg.value > 0, "Amount should be greater than 0");

balances[msg.sender] += msg.value;

emit Deposit(msg.sender, msg.value);

}

function withdraw(uint256 \_amt) public onlyOwner {

require(msg.sender.balance >= \_amt, "Insufficient balance");

balances[msg.sender] -= \_amt;

payable(msg.sender).transfer(\_amt);

emit Withdraw(msg.sender, \_amt);

}

function transfer(address \_to, uint256 \_amt) public {

require(\_to != address(0), "Invalid Address");

require(msg.sender.balance >= \_amt, "Insufficient balance");

balances[msg.sender] -= \_amt;

balances[\_to] += \_amt;

emit Transfer(msg.sender, \_to, \_amt);

}

function getBalance() public view returns(uint256) {

return balances[msg.sender];

}

function getBankWorth() public view returns(uint256) {

return address(this).balance;

}

}

**Output**

 

 