

BCT PRACTICAL 3

PS - Write a smart contract on a test network, for Bank account of a customer for following operations:

- Deposit money
- Withdraw Money
- Show balance

Solution-

Bank.sol

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

```
pragma solidity ^0.8.18;
```

```
contract Bank {
```

```
    // Mapping to store balances of each customer  
(address)
```

```
    mapping(address => uint256) private balances;
```

```
    // Event logs
```

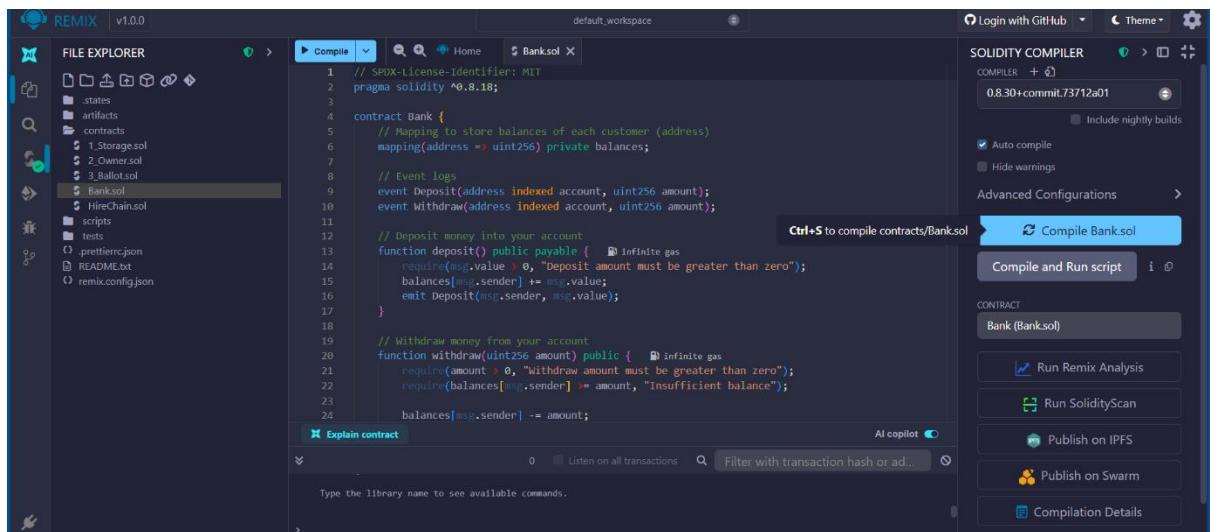
```
    event Deposit(address indexed account, uint256  
amount);
```

```
event Withdraw(address indexed account,  
uint256 amount);  
  
// Deposit money into your account  
function deposit() public payable {  
    require(msg.value > 0, "Deposit amount must  
be greater than zero");  
    balances[msg.sender] += msg.value;  
    emit Deposit(msg.sender, msg.value);  
}  
  
// Withdraw money from your account  
function withdraw(uint256 amount) public {  
    require(amount > 0, "Withdraw amount must  
be greater than zero");  
    require(balances[msg.sender] >= amount,  
    "Insufficient balance");  
  
    balances[msg.sender] -= amount;  
    payable(msg.sender).transfer(amount);
```

```
emit Withdraw(msg.sender, amount);  
}  
}
```

```
// Show your account balance  
function getBalance() public view returns  
(uint256) {  
    return balances[msg.sender];  
}  
}
```

OUTPUT-



The screenshot shows the REMIX IDE interface. The left sidebar is the 'FILE EXPLORER' showing files like Storage.sol, Owner.sol, Ballot.sol, Bank.sol, and HireChain.sol. The main area is the 'SOLIDITY COMPILER' window with the code for the 'Bank' contract. The code defines a mapping of addresses to uint256 balances, and two functions: 'deposit' and 'withdraw'. The 'deposit' function adds to the balance and emits a 'Deposit' event. The 'withdraw' function checks if the balance is sufficient and emits a 'Withdraw' event. The right sidebar has options for 'Auto compile', 'Compile and Run script', 'Run Remix Analysis', 'Run SolidityScan', 'Publish on IPFS', 'Publish on Swarm', and 'Compilation Details'.

```
// SPDX-License-Identifier: MIT  
pragma solidity ^0.8.18;  
  
contract Bank {  
    // Mapping to store balances of each customer (address)  
    mapping(address => uint256) private balances;  
  
    // Event logs  
    event Deposit(address indexed account, uint256 amount);  
    event Withdraw(address indexed account, uint256 amount);  
  
    // Deposit money into your account  
    function deposit() public payable {  
        require(msg.value > 0, "Deposit amount must be greater than zero");  
        balances[msg.sender] += msg.value;  
        emit Deposit(msg.sender, msg.value);  
    }  
  
    // Withdraw money from your account  
    function withdraw(uint256 amount) public {  
        require(amount > 0, "Withdraw amount must be greater than zero");  
        require(balances[msg.sender] >= amount, "Insufficient balance");  
        balances[msg.sender] -= amount;  
    }  
}
```

The screenshot shows the Remix IDE interface. On the left, the "DEPLOY & RUN TRANSACTIONS" sidebar is visible, showing an account balance of 0x5B3...eddC4 (100 ETH) and a gas limit of 3000000. The main workspace shows the source code for a Solidity contract named "Bank". The code defines a mapping of addresses to balances and includes deposit and withdraw functions. The right side of the interface contains the "SOLIDITY COMPILER" panel with compilation and deployment buttons, and a "CONTRACT" panel with options like "Run Remix Analysis" and "Publish on IPFS".

```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.18;
3
4 contract Bank {
5     mapping(address => uint256) private balances;
6
7     // Event logs
8     event Deposit(address indexed account, uint256 amount);
9     event Withdraw(address indexed account, uint256 amount);
10
11    // Deposit money into your account
12    function deposit() public payable {
13        require(msg.value > 0, "Deposit amount must be greater than zero");
14        balances[msg.sender] += msg.value;
15        emit Deposit(msg.sender, msg.value);
16    }
17
18    // Withdraw money from your account
19    function withdraw(uint256 amount) public {
20        require(amount > 0, "Withdraw amount must be greater than zero");
21        require(balances[msg.sender] >= amount, "Insufficient balance");
22
23        balances[msg.sender] -= amount;
24    }
}

```

This screenshot is nearly identical to the one above, showing the same Remix IDE interface with the Solidity contract "Bank.sol" open. The code remains the same, and the right-hand panels for compilation and deployment are visible.

This screenshot shows the Remix IDE after a transaction has been recorded. The "Transactions recorded" section indicates a single transaction was recorded. The "Deployed Contracts" section shows a deployed contract at address 0x9D7...B5E99 (MEMORY). The right-hand panels remain the same as in the previous screenshots.

REMX v1.0.0

default_workspace

Deploy & Run Transactions

Contract: Bank - contracts/Bank.sol

Value: 1 Ether

Transactions recorded: 1

Deployed Contracts: BANK AT 0x907...B5E99 (MEMORY)

Balance: 0 ETH

deposit withdraw uint256 amount getBalance

SOLIDITY COMPILER

Bank.sol

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

contract Bank {
    // Mapping to store balances of each address
    mapping(address => uint256) balances;
}
```

Explain contract AI copilot

Compile Home Bank.sol

Contract: Bank (Bank.sol)

Run Remix Analysis Run SolidityScan Publish on IPFS Publish on Swarm Compilation Details ABI Bytecode

REMX v1.0.0

default_workspace

Deploy & Run Transactions

Contract: Bank - contracts/Bank.sol

Deploy Publish to IPFS At Address Load contract from Address

Transactions recorded: 1

Deployed Contracts: BANK AT 0x907...B5E99 (MEMORY)

Balance: 0.9999999999999999 ETH

deposit withdraw 1 getBalance

Low level interactions CALLDATA

SOLIDITY COMPILER

Bank.sol

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

contract Bank {
    // Mapping to store balances of each address
    mapping(address => uint256) balances;
}
```

Explain contract AI copilot

Compile Home Bank.sol

Contract: Bank (Bank.sol)

Run Remix Analysis Run SolidityScan Publish on IPFS Publish on Swarm Compilation Details ABI Bytecode

REMX v1.0.0

default_workspace

Deploy & Run Transactions

Contract: Bank - contracts/Bank.sol

Deploy Publish to IPFS At Address Load contract from Address

Transactions recorded: 3

Deployed Contracts: BANK AT 0x907...B5E99 (MEMORY)

Balance: 0.9999999999999999 ETH

deposit withdraw 1 getBalance

Low level interactions

SOLIDITY COMPILER

Bank.sol

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

contract Bank {
    // Mapping to store balances of each address
    mapping(address => uint256) balances;
}
```

Explain contract AI copilot

Compile Home Bank.sol

Contract: Bank (Bank.sol)

Run Remix Analysis Run SolidityScan Publish on IPFS Publish on Swarm Compilation Details ABI Bytecode

BCT Prac 4

CODE

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity >=0.7.0 <0.9.0;

contract MarksManagmtSys {
    struct StudentStruct {
        uint ID;
        string fName;
        string lName;
        uint marks;
    }

    address public owner;
    uint public stdCount = 0;
    StudentStruct[] public stdRecords;

    constructor() {
        owner = msg.sender;
    }

    function addNewRecords(
        uint _ID,
        string memory _fName,
        string memory _lName,
        uint _marks
    ) public {
        stdCount++;
        stdRecords.push(StudentStruct(_ID, _fName, _lName, _marks));
    }
}
```

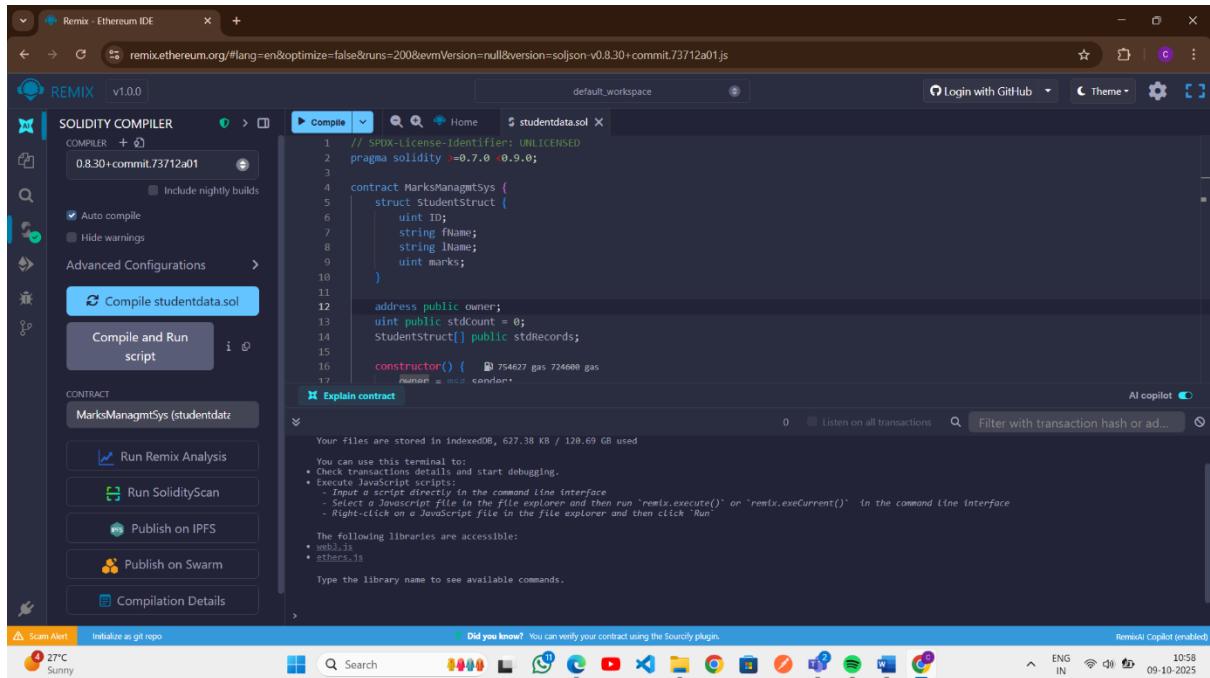
```
}

function getAllRecords() public view returns (StudentStruct[] memory) {
    StudentStruct[] memory records = new StudentStruct[](stdRecords.length);
    for (uint i = 0; i < stdRecords.length; i++) {
        records[i] = stdRecords[i];
    }
    return records;
}

// fallback function to handle calls to non-existing functions
fallback() external payable {
    // Called when no function matches
}

// receive function to handle plain ETH transfers
receive() external payable {
    // Accept ETH transfers
}
}
```

OUTPUT



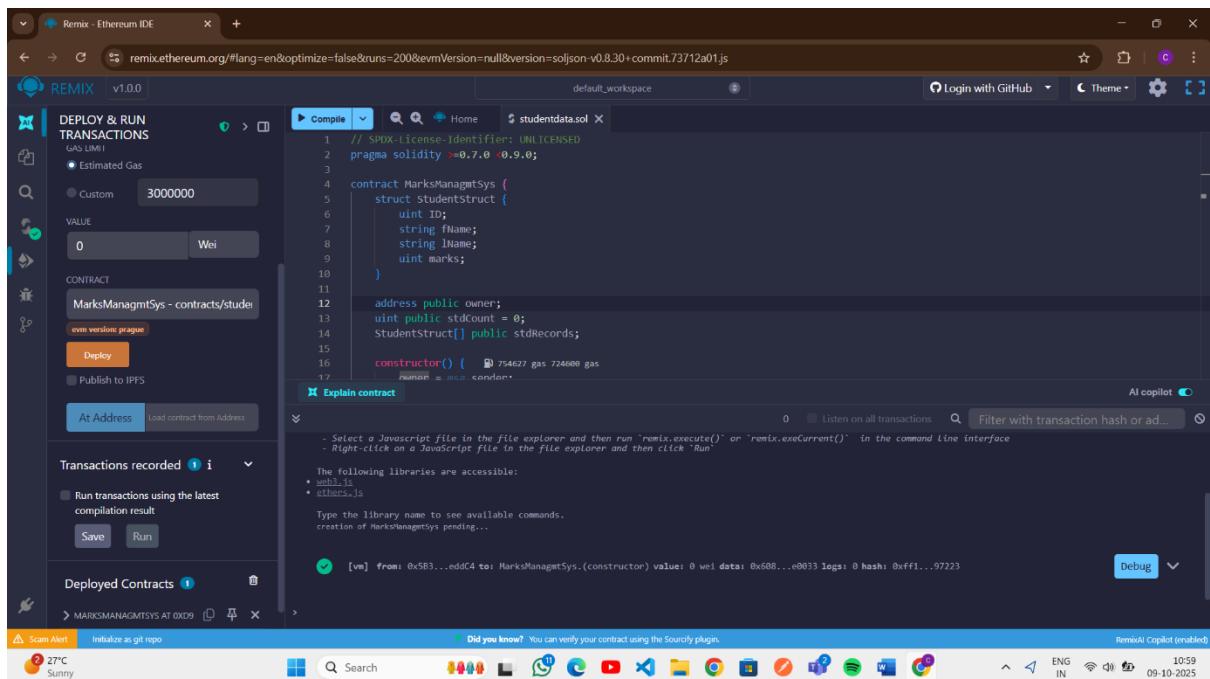
```
// SPDX-License-Identifier: UNLICENSED
pragma solidity >0.7.0 <0.9.0;

contract MarksManagementSystem {
    struct StudentStruct {
        uint ID;
        string fName;
        string lName;
        uint marks;
    }

    address public owner;
    uint public stdCount = 0;
    StudentStruct[] public stdRecords;

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

Compile studentdata.sol



```
// SPDX-License-Identifier: UNLICENSED
pragma solidity >0.7.0 <0.9.0;

contract MarksManagementSystem {
    struct StudentStruct {
        uint ID;
        string fName;
        string lName;
        uint marks;
    }

    address public owner;
    uint public stdCount = 0;
    StudentStruct[] public stdRecords;

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

Deploy studentdata.sol

The screenshot shows the Remix Ethereum IDE interface. In the top right, there's a GitHub integration button and a theme switcher. The main area has tabs for 'REMX' and 'studentdata.sol'. The code editor contains the Solidity source code for the `MarksManagementSys` contract. Below the code, the 'DEPLOY & RUN TRANSACTIONS' sidebar shows a deployed contract at address `0xD91...39138`. It lists two student records: Tejaswini (ID 1) and Chandargi (ID 2). The 'ADDNEWRECORDS' section shows a transaction being prepared to add a new record with ID 3, name Chandargi, and marks 95. The transaction details show it will cost 754627 gas and 7246000 wei. The transaction is pending, with its hash visible in the logs.

Function addNewRecords to add students data

This screenshot is similar to the previous one but focuses on the `getAllRecords` function. The sidebar shows the function is selected. The transaction details now show a call to `MarksManagementSys.getAllRecords()`, which costs 754627 gas and 7246000 wei. The transaction is pending, with its hash visible in the logs.

Function getAllRecords to fetch all students data

The screenshot shows the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar lists a deployed contract named 'MARKSMANAGMTSYS' at address 0xD91...39138. It shows a balance of 0 ETH and two functions: 'addNewRecords' and 'getAllRecords'. The 'owner' field is also listed. On the right, the code editor displays the Solidity source code for the 'MarksManagement' contract. The 'Explain contract' panel below the code provides detailed information about the 'stdCount' function, including its call trace, parameters, and return value.

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity >0.7.0 <0.9.0;

contract MarksManagement {
    struct StudentStruct {
        uint ID;
        string fName;
        string lName;
        uint marks;
    }

    address public owner;
    uint public stdCount = 0;
    StudentStruct[] public stdRecords;

    constructor() {
        owner = msg.sender;
    }

    function addNewRecords(uint256 _ID, string memory _fName, string memory _lName, uint256 _marks) external {
        stdCount++;
        stdRecords.push(StudentStruct(_ID, _fName, _lName, _marks));
    }

    function getAllRecords() external view returns (StudentStruct[] memory) {
        return stdRecords;
    }
}
```

Fetch Owner

This screenshot is similar to the previous one but focuses on the 'stdCount' function. The 'Explain contract' panel is expanded to show the details of this specific function, including its parameters ('from' and 'to'), execution cost (2491 gas), input (0xfc6...15f2a), output (0x0001), decoded input ({}), decoded output ({"q": "uint256: 1"}), and logs ({}).

```
function stdCount() external view returns (uint) {
    return stdCount;
}
```

Function to get the number of student entries

