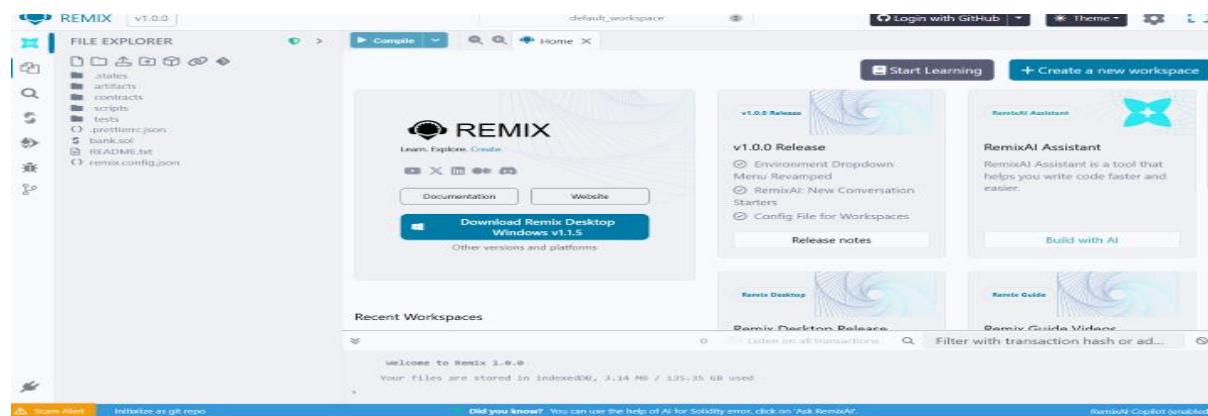


ASSIGNMENT NO:04

AIM: Write a program in solidity to create Student data. Use the following constructs:

- Structures
- Arrays
- Fallback Deploy this as smart contract on Ethereum and Observe the transaction fee and Gas values.



This screenshot shows the Remix IDE interface after deployment. The 'DEPLOY & RUN TRANSACTIONS' sidebar indicates 'Transactions recorded: 1'. The 'Deployed Contracts' section shows a deployed contract named 'STUDENTDATA' at address 0x00FC...9ABF with a balance of 0 ETH. It lists four functions: 'addStudent', 'getStudent', 'getStudentCount', and 'isStudentIdentified', each with their respective parameters and return types. The main workspace shows the Solidity code for the 'StudentData' contract. The code defines a public owner and a private struct 'Student' with fields id, name, course, and marks. An array 'students' holds all student instances. The 'Explain contract' tool is open, providing documentation for the code. The bottom status bar includes 'Scan Alert' (Initialize as git repo), 'Did you know?' (Ask RemixAI), and 'RemixAI Copilot (disabled)'.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.8;

/*
 * @title StudentData
 * @notice Contract to store and manage student data with access control, existence check, and fallback
 */
contract StudentData {
    address public owner;

    /**
     * @dev Struct to hold student information
     */
    struct Student {
        uint id;
        string name;
        string course;
        uint256 marks;
    }

    // Array to store all students
    Student[] public students;
}
```

This screenshot shows the complete Solidity code for the 'StudentData' contract in the Remix IDE. The code includes a mapping to track student IDs, an event for fallback, and a constructor that sets the deployer as the owner. It also includes a modifier 'onlyOwner' that checks if the sender is the owner. The 'Explain contract' tool is open, providing detailed explanations for the code. The bottom status bar includes 'Scan Alert' (Initialize as git repo), 'Did you know?' (Ask RemixAI), and 'RemixAI Copilot (disabled)'.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.8;

// Mapping to track if a student ID already exists.
mapping(uint => bool) private studentExists;

// Event for fallback
eventFallbackCalled(address sender, uint value, string message);

// Set the contract deployer as the owner
constructor() {
    _msgSender();
    owner = msg.sender;
}

/**
 * @dev Modifier to allow only the owner to perform certain actions.
 */
modifier onlyOwner() {
    require(msg.sender == owner, "Only the owner can perform this action");
}
```

The screenshot shows the REMIX IDE interface with the `StudentRegister.sol` contract loaded. The sidebar displays deployed contracts and their interactions. The main area shows the Solidity code for the `StudentRegister` contract, which includes functions for adding students, getting student details by index, checking if a student exists by ID, and a fallback function for receiving ETH.

```
44  */
45  * @notice Add a new student, only if they don't already exist
46  * @param _id Unique student ID
47  * @param _name Student name
48  * @param _course Student course
49  * @param _marks Student marks
50  */
51 function addStudent(uint _id, string memory _name, string memory _course, uint256 _marks) public onlyOwner
52     require(!studentExists[_id], "Student already exists");
53
54     students.push(Student(_id, _name, _course, _marks));
55     studentExists[_id] = true;
56
57     emit StudentAdded(_id, _name);
58
59 /**
60  * @notice Get the total number of students
61  */
62 function getStudentCount() public view returns (uint) {
63     return students.length;
64 }
```

The screenshot shows the REMIX IDE interface with the `StudentRegister.sol` contract loaded. The main area shows the Solidity code for the `StudentRegister` contract, which includes functions for getting student count, getting student details by index, and checking if a student exists by ID.

```
63 function getStudentCount() public view returns (uint) {
64     return students.length;
65 }
66 /**
67  * @notice Get student details by array index
68  */
69 function getStudent(uint index) public view returns (uint, string memory, string memory, uint256) {
70     require(index < students.length, "Invalid index");
71
72     Student memory s = students[index];
73     return (s.id, s.name, s.course, s.marks);
74 }
75
76 /**
77  * @notice Check if a student exists by ID
78  */
79 function isStudentExist(uint _id) public view returns (bool) {
80     return studentExists[_id];
81 }
82
83 /**
84  * @notice Fallback function for invalid calls or direct ETH sends
85  */
86 fallback() external payable {
87     emit FallbackCalled(msg.sender, msg.value, "Fallback function was triggered");
88 }
89
90 /**
91  * @notice Receive ETH directly
92  */
93 receive() external payable {}
```

The screenshot shows the REMIX IDE interface with the `StudentRegister.sol` contract loaded. The main area shows the Solidity code for the `StudentRegister` contract, which includes a fallback function for receiving ETH directly.

```
81     return studentExists[_id];
82 }
83
84 /**
85  * @notice Fallback function for invalid calls or direct ETH sends
86  */
87 fallback() external payable {
88     emit FallbackCalled(msg.sender, msg.value, "Fallback function was triggered");
89 }
90
91 /**
92  * @notice Receive ETH directly
93  */
94 receive() external payable {}
```

Deploy:

The screenshot shows the Remix IDE interface. On the left, the sidebar displays "DEPLOY & RUN TRANSACTIONS" with a list of deployed contracts under "STUDENTDATA AT 0xae0...96bf". One contract, "addStudent", is highlighted. The main workspace shows the "Compiled" code for the "StudentRegister.sol" contract. The code includes functions like `addStudent`, `getStudent`, and `isStudentExist`. The "Explain contract" section at the bottom provides transaction details for a recent call to `addStudent`.

```
81     return studentExists[_id];
82   }
83
84   /**
85    * @notice Fallback function for invalid calls or direct ETH sends
86    */
87   fallback() external payable {
88     emit FallbackCalled(msg.sender, msg.value, "Fallback function was triggered");
89   }
90
91   /**
92    * @notice Receive ETH directly
93    */
94   receive() external payable {}
```

VM Call Details: [vm] from: 0x5B3...eddC4 to: StudentData.addStudent(uint256,string,string,uint256) 0xaE0...9688b value: 0 wei data: 0x531...00000 logs: 1 hash: 0xeb2...d7a81

Add Student:

The screenshot shows the Remix IDE interface after adding a new student record. The "DEPLOY & RUN TRANSACTIONS" sidebar now lists two entries under "STUDENTDATA AT 0xae0...96bf": "addStudent" and "getStudent". The "addStudent" entry shows parameters: 0: uint256; 1: string; 2: string; 3: uint256. The "getStudent" entry shows a result of 0. The main workspace shows the same Solidity code as before, with the "Explain contract" section now detailing a call to `getStudent`.

```
81     return studentExists[_id];
82   }
83
84   /**
85    * @notice Fallback function for invalid calls or direct ETH sends
86    */
87   fallback() external payable {
88     StructuredDocumentation undefined gas
89     emit FallbackCalled(msg.sender, msg.value, "Fallback function was triggered");
90   }
91
92   /**
93    * @notice Receive ETH directly
94    */
95   receive() external payable {}
```

VM Call Details: [call] from: 0x5B38Daa701c568545dCfcB03FcB875f56beddC4 to: StudentData.getStudent(uint256) data: 0x642...00000

Get Student Data:

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar shows a balance of 0 ETH and two transactions: 'addStudent' and 'getStudent'. The 'getStudent' transaction has a value of 0 and a recipient of '0x531...00000'. The 'Explain contract' section on the right shows the Solidity code for the 'getStudent' function and a list of transaction logs. The logs include:

- [vm] from: 0x503...eddca to: StudentData.(constructor) value: 0 wei data: 0x600...80000 logs: 0
- [vm] from: 0x503...eddca to: StudentData.addStudent(uint256,string,string,uint256) 0xa0...9688b value: 0 wei data: 0x531...00000 logs: 1 hash: 0xeb2...d7a81
- [vm] from: 0x503...eddca to: StudentData.addStudent(uint256,string,string,uint256) 0xa0...9688b value: 0 wei data: 0x531...00000 logs: 1 hash: 0xeb2...d7a81
- [call] from: 0x503...eddca to: StudentData.getStudent(uint256) 0xa0...9688b data: 0x6a2...00000
- [call] from: 0x503...eddca to: StudentData.getStudentCount() 0xa0...9688b data: 0x41e...8c407

Get Total No.of Students:

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar shows a balance of 0 ETH and three transactions: 'addStudent', 'getStudent', and 'isStudentExist'. The 'isStudentExist' transaction has a value of 1 and a recipient of '0x531...00000'. The 'getStudent' transaction has a value of 0 and a recipient of '0x531...00000'. The 'Explain contract' section on the right shows the Solidity code for the 'isStudentExist' function and a list of transaction logs. The logs include:

- [vm] from: 0x503...eddca to: StudentData.addStudent(uint256,string,string,uint256) 0xa0...9688b value: 0 wei data: 0x531...00000 logs: 1 hash: 0xeb2...d7a81
- [vm] from: 0x503...eddca to: StudentData.addStudent(uint256,string,string,uint256) 0xa0...9688b value: 0 wei data: 0x531...00000 logs: 1 hash: 0xeb2...d7a81
- [call] from: 0x503...eddca to: StudentData.getStudent(uint256) 0xa0...9688b data: 0x6a2...00000
- [call] from: 0x503...eddca to: StudentData.getStudentCount() 0xa0...9688b data: 0x41e...8c407
- [call] from: 0x503...eddca to: StudentData.isStudentExist(uint256) 0xa0...9688b data: 0xe90...00001

Check Student Exist:

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar has 'STUDENTDATA AT 0x4E0...96Bf' selected. Under 'addStudent', the parameters are set to 2, "Bob", "Solidity", 90. Under 'getStudent', the student ID is 0. In the main pane, the Solidity code for the `isStudentExist` function is displayed:

```
81     }
82     }
83     }
84     /**
85      * @notice Fallback function for invalid calls or direct ETH sends
86      */
87 }
```

The 'Explain contract' section shows the following interactions:

- [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: StudentData.getStudent(uint256) data: 0x642...00000
- [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: StudentData.getStudentCount() data: 0x41e...0c407
- [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: StudentData.isStudentExist(uint256) data: 0x640...00001
- [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: StudentData.isStudentExist() data: 0x640...00005

At the bottom, the status bar says: Did you know? You can use the help of AI for Solidity error, click on 'Ask RemixAI'.

Check Owner :

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' sidebar has 'STUDENTDATA AT 0x4E0...96Bf' selected. Under 'addStudent', the parameters are set to 2, "Bob", "Solidity", 90. Under 'getStudent', the student ID is 0. In the main pane, the Solidity code for the `isStudentExist` function is displayed:

```
81     }
82     }
83     }
84     /**
85      * @notice Fallback function for invalid calls or direct ETH sends
86      */
87 }
```

The 'Explain contract' section shows the following interactions:

- [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: StudentData.getStudent(uint256) data: 0x642...00000
- [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: StudentData.getStudentCount() data: 0x41e...0c407
- [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: StudentData.isStudentExist(uint256) data: 0x640...00001
- [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: StudentData.isStudentExist() data: 0x640...00005
- [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: StudentData.owner() data: 0x8da...5cb5b

At the bottom, the status bar says: Did you know? You can use the help of AI for Solidity error, click on 'Ask RemixAI'.

The screenshot shows the Remix IDE interface. On the left, there's a sidebar titled "DEPLOY & RUN TRANSACTIONS" with buttons for "addStudent" (with parameters 2, "Bob", "Solidity", 90) and "getStudent" (with parameter 0). Below these are buttons for "getStudentCount" (parameter 0), "isStudentExist" (parameter 5), "owner" (parameter 0), and "students" (parameter 1). The main area displays the Solidity code for the `StudentRegister` contract:

```
81     }
82 }
83 
84 /**
85  * @notice Fallback function for invalid calls or direct ETH sends
86 */
87 }
```

Below the code, the "Explain contract" section provides details about the interactions:

- call to `StudentData.getStudentCount`
- call to `StudentData.isStudentExist`
- call to `StudentData.isStudentExist(uint256)`
- call to `StudentData.owner`
- call to `StudentData.students`

Each interaction has a "Debug" button next to it. At the bottom of the interface, there are buttons for "Scan ABI", "Initialize git repo", and "Did you know? You can use the help of AI for Solidity error, click on 'Ask RemixAI'". The status bar at the bottom right indicates "RemixAI Copilot (enabled)".