

Answer	Coding Efficiency	Viva	Timely Completion	Total	Dated Sign of Subject Teacher
5	5	5	5	20	

Expected Date of Completion:..... Actual Date of Completion:.....

Group C

Assignment No: 4

Title of the Assignment: 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.

Objective of the Assignment: Students should be able to learn about Solidity. Its datatypes and implementations.

Prerequisite:

1. Basic Programming Logic
2. Basic knowledge of Solidity

Contents for Theory:

1. Solidity - Arrays
 2. Solidity - Structures
 3. Solidity – Fallback
 4. Implementation
-

SNJB's Late Sau.K.B.Jain College of Engineering,Chandwad

1. Solidity – Arrays :

Arrays are data structures that store the fixed collection of elements of the same data types in which each and every element has a specific location called index. Instead of creating numerous individual variables of the same type, we just declare one array of the required size and store the elements in the array and can be accessed using the index. In Solidity, an array can be of fixed size or dynamic size. Arrays have a continuous memory location, where the lowest index corresponds to the first element while the highest represents the last.

Creating an Array

To declare an array in Solidity, the data type of the elements and the number of elements should be specified. The size of the array must be a positive integer and data type should be a valid Solidity type

Syntax:

```
<data type> <array name>[size] = <initialization>
```

Fixed-size Arrays :

The size of the array should be predefined. The total number of elements should not exceed the size of the array. If the size of the array is not specified then the array of enough size is created which is enough to hold the initialization.

Dynamic Array:

The size of the array is not predefined when it is declared. As the elements are added the size of array changes and at the runtime, the size of the array will be determined.

Array Operations :**1. Accessing Array Elements:**

The elements of the array are accessed by using the index. If you want to access i th element then you have to access $(i-1)$ th index.

2. Length of Array:

Length of the array is used to check the number of elements present in an array. The size of the memory array is fixed when they are declared, while in case the dynamic array is defined at runtime so for manipulation length is required.

3. Push:

Push is used when a new element is to be added in a dynamic array. The new element is always added at the last position of the array.

4. Pop:

Pop is used when the last element of the array is to be removed in any dynamic array.

2. Solidity – Structures :

Structs in Solidity allows you to create more complicated data types that have multiple properties. You can define your own type by creating a **struct**.

They are useful for grouping together related data.

Structs can be declared outside of a contract and imported in another contract. Generally, it is used to represent a record. To define a structure *struct* keyword is used, which creates a new data type.

Syntax:

```
struct <structure_name> {  
    <data type> variable_1;  
    <data type> variable_2;  
}
```

For accessing any element of the structure, 'dot operator' is used, which separates the struct variable and the element we wish to access. To define the variable of structure data type structure name is used.

3. Solidity – Fallback :

The solidity fallback function is executed if none of the other functions match the function identifier or no data was provided with the function call. Only one unnamed function can be assigned to a contract and it is executed whenever the contract receives plain Ether without any data. To receive Ether and add it to the total balance of the contract, the fallback function must be marked payable. **If no such function exists, the contract cannot receive Ether through regular transactions and will throw an exception.**

Properties of a fallback function:

1. Has no name or arguments.
2. If it is not marked **payable**, the contract will throw an exception if it receives plain ether without data.
3. Can not return anything.
4. Can be defined once per contract.
5. It is also executed if the caller meant to call a function that is not available

6. It is mandatory to mark it external.
7. It is limited to 2300 gas when called by another function. It is so for as to make this function call as cheap as possible.

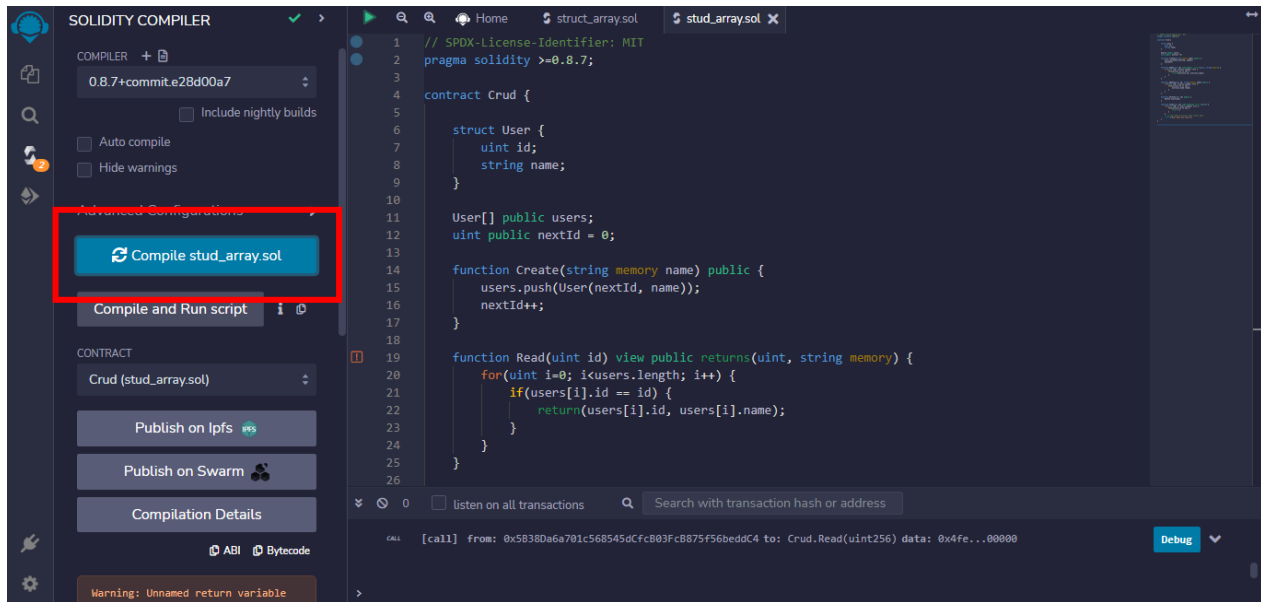
4. Implementation

Code -

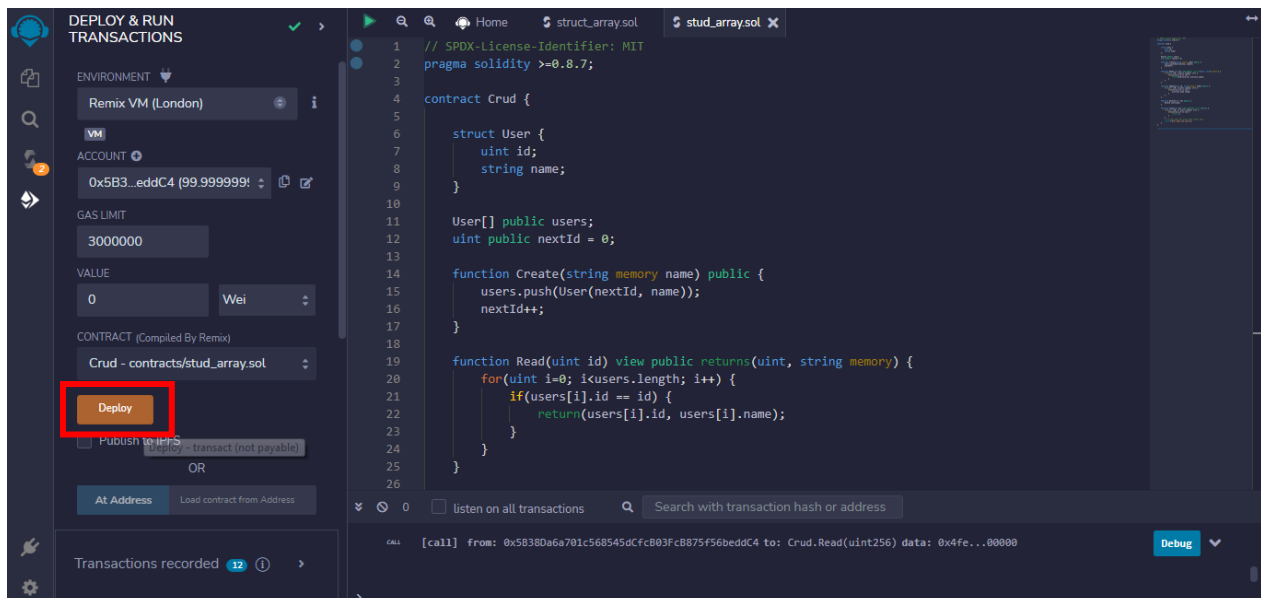
```
pragma solidity ^0.5.0;
contract Crud {
    struct User {
        uint id;
        string name;
    }
    User[] public users;
    uint public nextId = 0;
    function Create(string memory name) public {
        users.push(User(nextId, name));
        nextId++;
    }
    function Read(uint id) view public returns(uint, string memory) {
        for(uint i=0; i<users.length; i++) {
            if(users[i].id == id) {
                return(users[i].id, users[i].name);
            }
        }
    }
    function Update(uint id, string memory name) public {
        for(uint i=0; i<users.length; i++) {
            if(users[i].id == id) {
                users[i].name =name;
            }
        }
    }
    function Delete(uint id) public {
        delete users[id];
    }
    function find(uint id) view internal returns(uint) {
        for(uint i=0; i< users.length; i++) {
            if(users[i].id == id) {
                return i;
            }
        }
        // if user does not exist then revert back
        revert("User does not exist");
    }
}
```

Output –

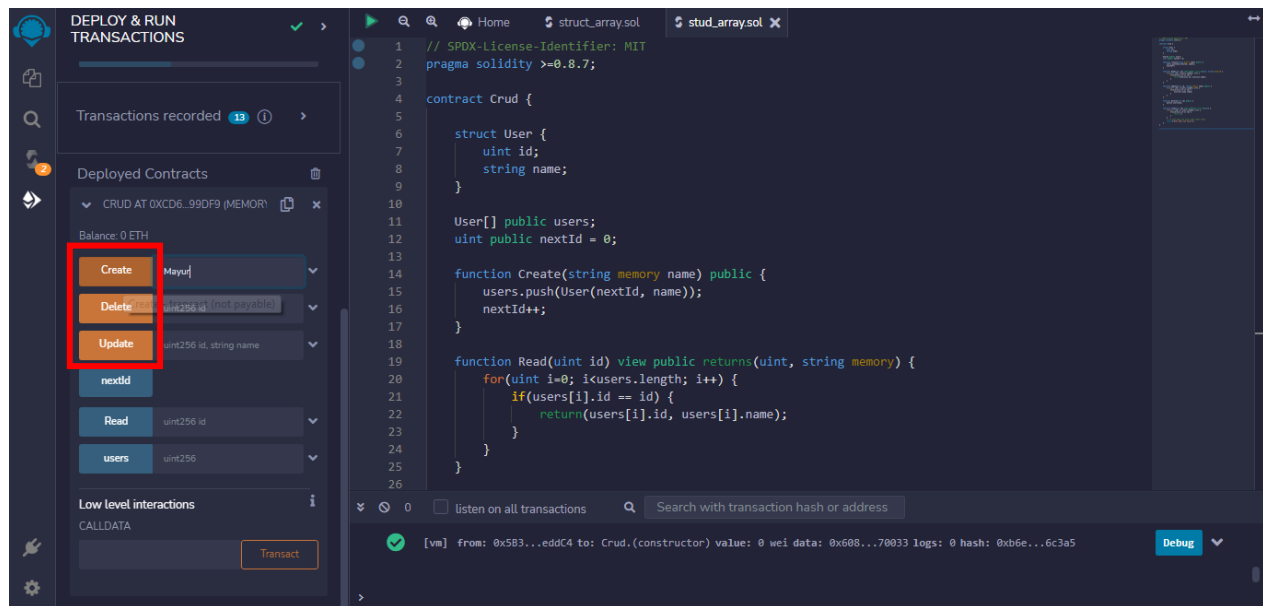
Step 1 – Compile the program by clicking on compile button.



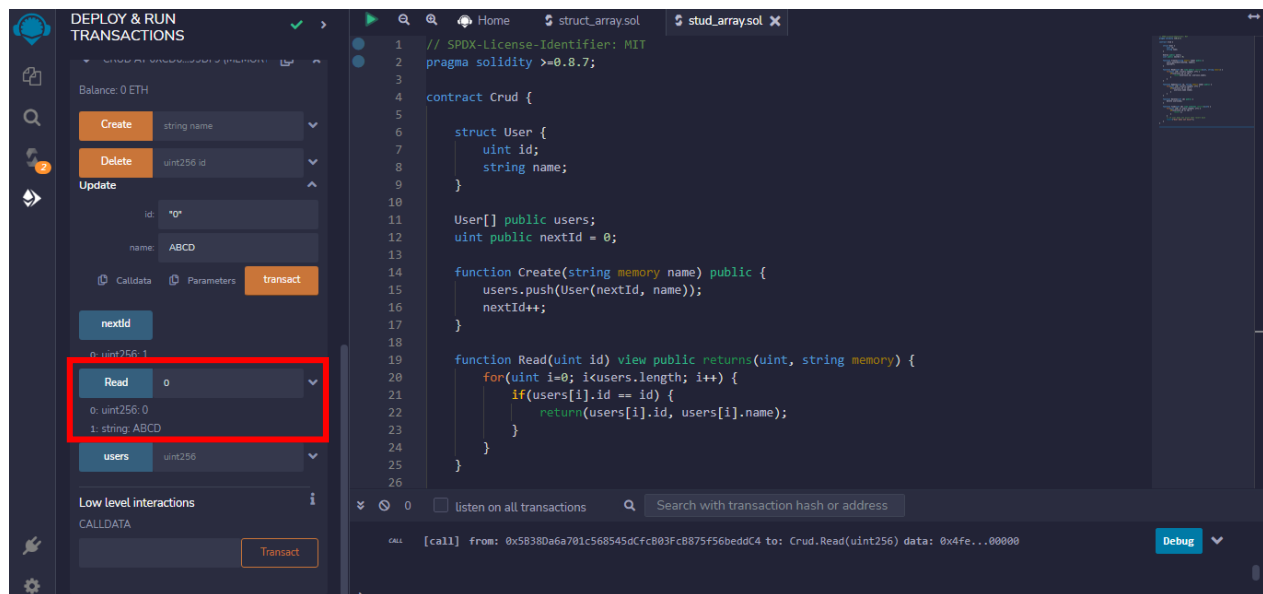
Step 2 – After the Successful compilation, Deploy the contract to see the output.



Step 3 - Now you can check the output. You can insert, update and delete the student data using your smart contract.



Step 4 – After entering your data, you can read the data using ID.



Conclusion-

In this way we have created array, structure and used fallback function in solidity.

Assignment Question:

- 1. What is fixed array and dynamic array in solidity?**
- 2. What is Array in solidity ?**
- 3. What is structure in solidity? Define its syntax.**
- 4. What is fallback function ?**

Reference link

- <https://www.geeksforgeeks.org/solidity-arrays/?ref=lbp>
- tutorialspoint.com/solidity/solidity_structs.htm