Practical No 4.

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.

Objective: Understand and explore the working of Blockchain Technology and its Application

Course Outcome:

CO6: Interpret the basic concepts in Blockchain Technology and its Application.

Theory:

Differences to Layout in Storage

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>

The following array occupies 32 bytes (1 slot) in storage, but 128 bytes (4 items with 32 bytes each) in memory.

```
uint8[4] a;
```

Defining a Struct

To define a Struct, you must use the **struct** keyword. The struct keyword defines a new data type, with more than one member. The format of the struct statement is as follows −

```
struct struct_name {
   type1 type_name_1;
   type2 type_name_2;
   type3 type_name_3;
}
```

Example Struct Layout

The following struct occupies 96 bytes (3 slots of 32 bytes) in storage, but 128 bytes (4 items with 32 bytes each) in memory.

```
struct S {
   uint a;
   uint b;
   uint8 c;
   uint8 d;
}
```

Enums:

Enums restrict a variable to have one of only a few predefined values. The values in this enumerated list are called enums.

With the use of enums it is possible to reduce the number of bugs in your code.

For example, if we consider an application for a fresh juice shop, it would be possible to restrict the glass size to small, medium, and large. This would make sure that it would not allow anyone to order any size other than small, medium, or large.

Example:

```
enum FreshJuiceSize{ SMALL, MEDIUM, LARGE }
```

Mapping:

Mapping is a reference type as arrays and structs. Following is the syntax to declare a mapping type.

```
mapping(_KeyType => _ValueType)
```

Where

- **_KeyType** − can be any built-in types plus bytes and string. No reference type or complex objects are allowed.
- **_ValueType** − can be any type.

Considerations

- Mapping can only have type of **storage** and are generally used for state variables.
- Mapping can be marked public. Solidity automatically create getter for it.

String:

Solidity supports String literal using both double quote (") and single quote ('). It provides string as a data type to declare a variable of type String.

```
pragma solidity ^0.5.0;

contract SolidityTest {
   string data = "test";
}
```

Solidity fallback function :

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.
In Solidity, a fallback function is **an external function with neither a name, parameters, or return values**.
It is executed in one of the following cases:

- ➔ If a function identifier doesn't match any of the available functions in a smart contract.
- ➔ If there was no data supplied along with the function callOnly 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.

.

How do you trigger a fallback function Solidity?

The solidity fallback function is executed **if no one of the opposite functions matches the function identifier**. It is executed if no data was given the call. Just one unnamed function is frequently assigned to a contract. It's executed whenever the contract receives plain Ether with no data.

Conclusion:

Printout of Program code

Output screen shot paste below