

ASSIGNMENT 2 BT

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;
contract marksmanagement{
    struct student{
        int id;
        string fname;
        string lname;
        int marks;
    }
    uint public stdcount=0;
    address owner;

    mapping (int =>student) public std;
    modifier onlyOwner(){
        require(owner==msg.sender);
        _;
    }
    constructor(){
        owner=msg.sender;
    }
    function addnewRecords(int id,string memory fname,string memory lname,int marks) public
    onlyOwner{
        stdcount=stdcount+1;
        std[id]=student(id,fname,lname,marks);

    }

}
```

This code is another Solidity smart contract. Let's break it down in simple terms:

1. ``pragma solidity >=0.7.0 <0.9.0;``: This line specifies the version of Solidity that the contract is compatible with.
2. ``contract marksmanagement {``: This line defines a smart contract called "marksmanagement."
3. ``struct student { ... }``: This defines a custom data structure called "student," which has four properties: an ID (integer), a first name (string), a last name (string), and marks (integer). This structure is used to represent student records.
4. ``uint public stdcount = 0;``: This declares a public unsigned integer variable "stdcount" and initializes it to 0. It's used to keep track of the number of student records.
5. ``address owner;``: This declares a variable "owner" to store the Ethereum address of the contract owner.
6. ``mapping (int => student) public std;``: This is a mapping data structure that associates student IDs (integers) with their corresponding student records. It's public, so anyone can read the student records.
7. ``modifier onlyOwner() { ... }``: This is a custom modifier called "onlyOwner." It ensures that only the contract owner can perform certain functions. It checks whether the sender of the transaction (`msg.sender`) matches the contract owner's address.
8. ``constructor() { ... }``: This is the constructor function that runs only once when the contract is created. It sets the "owner" to the address that deploys the contract.
9. ``function addnewRecords(int id, string memory fname, string memory lname, int marks) public onlyOwner { ... }``: This function allows the owner to add new student records. It takes as input the student's ID, first name, last name, and marks. It increments the "stdcount" to keep track of the number of records and stores the student's information in the "std" mapping.

In summary, this smart contract is used for managing student records. The owner can add new student records, and these records are stored in a mapping where each student's ID is associated with their details. The contract also keeps track of the total number of records. The "onlyOwner" modifier ensures that only the owner can add new records to the contract.

QUESTIONS

Certainly, here are some viva questions for the code you provided, along with simple answers:

1. What does this Solidity smart contract do?

- Answer: This Solidity smart contract is used for managing student records.

2. What is the purpose of the "stdcount" variable?

- Answer: The "stdcount" variable keeps track of the number of student records in the contract.

3. How is the owner of the contract determined?

- Answer: The owner is determined when the contract is deployed and is set to the Ethereum address that deploys the contract.

4. What does the "onlyOwner" modifier do?

- Answer: The "onlyOwner" modifier ensures that only the contract owner can perform certain functions, like adding new student records.

5. What data structure is used to store student records, and how is it defined?

- Answer: Student records are stored in a mapping, and the structure of each student record is defined using a custom "student" struct.

6. How can the owner add a new student record?

- Answer: The owner can add a new student record by calling the "addnewRecords" function and providing the student's ID, first name, last name, and marks as arguments.

7. Can anyone read the student records in this contract?

- Answer: Yes, anyone can read the student records because the "std" mapping is declared as public.

8. What happens when a new student record is added?

- Answer: When a new student record is added, the "stdcount" is incremented, and the student's information is stored in the "std" mapping.

These questions and answers should help you understand the key aspects of the code and how it functions.