PRACTICAL 4->

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract StudentData {

    // Owner of the contract

    address public owner;

    // Structure to store student details

    struct Student {

        string name;

        uint age;

        string course;

        uint marks;

        bool isExist;

    }

    // Dynamic array of students

    Student[] public students;

    // Event to log the addition of a new student

    event StudentAdded(string name, uint age, string course, uint marks);

    // Modifier to allow only the contract owner to add students

    modifier onlyOwner() {

        require(msg.sender == owner, "Only the owner can perform this action.");

        \_;

    }

    // Constructor to initialize the owner of the contract

    constructor() {

        owner = msg.sender;

    }

    function addStudent(string memory \_name, uint \_age, string memory \_course, uint \_marks) public onlyOwner {

        // Add the new student to the array

        students.push(Student(\_name, \_age, \_course, \_marks, true));

        emit StudentAdded(\_name, \_age, \_course, \_marks);

    }

    function getStudent(uint index) public view returns (string memory name, uint age, string memory course, uint marks) {

        require(index < students.length, "Student does not exist.");

        Student storage student = students[index];

        return (student.name, student.age, student.course, student.marks);

    }

    receive() external payable {

        // Logic for receiving Ether can be added if needed

    }

    fallback() external payable {

        // Fallback logic can be added if needed

    }

    function getContractBalance() public view returns (uint) {

        return address(this).balance;

    }

}



