1)

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract ArrayExample {

uint[] public numbers;

uint public sum;

// Function to add numbers to the array

function addNumbers(uint[] memory \_numbers) public {

numbers = \_numbers;

}

// Function to display the array

function getNumbers() public view returns (uint[] memory) {

return numbers;

}

// Function to calculate the sum of numbers in the array

function calculateSum() public {

sum = 0;

for (uint i = 0; i < numbers.length; i++) {

sum += numbers[i];

}

}

}

2)

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract BankAccount {

address public owner;

mapping(address => uint256) public balances;

event Deposit(address indexed account, uint256 amount);

event Withdrawal(address indexed account, uint256 amount);

modifier onlyOwner() {

require(msg.sender == owner, "Only the owner can call this function");

\_;

}

constructor() {

owner = msg.sender;

}

function deposit() external payable {

require(msg.value > 0, "Deposit amount must be greater than 0");

balances[msg.sender] += msg.value;

emit Deposit(msg.sender, msg.value);

}

function withdraw(uint256 amount) external {

require(amount > 0, "Withdrawal amount must be greater than 0");

require(balances[msg.sender] >= amount, "Insufficient balance");

balances[msg.sender] -= amount;

payable(msg.sender).transfer(amount);

emit Withdrawal(msg.sender, amount);

}

function getBalance() external view returns (uint256) {

return balances[msg.sender];

}

}

3)

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract StudentData {

struct Student {

uint256 id;

string name;

uint256 age;

}

Student[] public students;

event StudentAdded(uint256 indexed id, string name, uint256 age);

// Fallback function to accept Ether

receive() external payable {

// Fallback function logic (if needed)

}

function addStudent(uint256 \_id, string memory \_name, uint256 \_age) public {

students.push(Student(\_id, \_name, \_age));

emit StudentAdded(\_id, \_name, \_age);

}

function getStudentCount() public view returns (uint256) {

return students.length;

}

function getStudent(uint256 index) public view returns (uint256, string memory, uint256) {

require(index < students.length, "Index out of bounds");

return (students[index].id, students[index].name, students[index].age);

}

}