// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract BankAccount {

// State variable to store the balance of each account

mapping(address => uint256) private balances;

// Event to log deposits

event Deposit(address indexed account, uint256 amount);

// Event to log withdrawals

event Withdraw(address indexed account, uint256 amount);

// Function to deposit money into the account

function deposit(uint256 amount) public payable {

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

// Ensure the value sent with the transaction matches the amount specified

require(msg.value == amount, "Amount does not match the value sent");

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

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

}

// Function to withdraw money from the account

function withdraw(uint256 amount) public {

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

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

balances[msg.sender] -= amount;

// Transfer the amount back to the caller

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

emit Withdraw(msg.sender, amount);

}

// Function to check the balance of the account

function getBalance() public view returns (uint256) {

return balances[msg.sender];

}

}

# Input from the user

n = int(input("Enter the number of terms in the Fibonacci series: "))

def calculate\_fibonacci\_series(n):

a, b = 0, 1

step\_count = 0

fibonacci\_series = []

for i in range(n):

step\_count += 1

fibonacci\_series.append(a)

a, b = b, a+b

return fibonacci\_series, step\_

# Calculate the Fibonacci series and step count

fibonacci\_series, step\_count = calculate\_fibonacci\_series(n)

print(f"Fibonacci Series for the first {n} terms: {fibonacci\_series}")

print(f"Step count: {step\_count}")