//SPDX-License-Identifier: MIT

pragma solidity ^0.5.7;

// Essentially, a smart contract can be thought of as a programmable

// intermediary which can facilitate transactions between parties

// and autonomously settling disputes. Which is perfect for something

// like a 'Last Will And Testament' to distribute an inheritance

// to various beneficiaries.

// EXERCISE:

// ---------

// 1. refactor the code to use an array of Structs to hold the beneficiary information,

// without using a mapping.

// 2. create a function to translate the Value entered in Ether to Wei.

// 3. Add some require statement(s) to make sure the balance in the contract is sufficient to

// cover the setInheritance values.

contract LastWillAndTestament {

address owner;

uint funds;

bool isDeceased;

constructor() public payable {

owner = msg.sender;

funds = msg.value;

isDeceased = false;

}

modifier onlyOwner() {

require(msg.sender==owner, "You are not the owner of the contract.");

\_;

}

modifier isOwnerDeceased() {

require(isDeceased==true, "Contract owner must be deceased for funds to be distributed.");

\_;

}

// This emulates 'iterating over a mapping' which cannot be done directly.

// Here we are iterating over an array of keys

// to plug into the mapping to get the associated value.

address payable[] beneficiaryAccounts;

mapping(address=>uint) inheritance;

function setInheritance(address payable \_account, uint \_inheritAmt) public onlyOwner {

beneficiaryAccounts.push(\_account);

inheritance[\_account] = \_inheritAmt;

}

function distributeFunds() private isOwnerDeceased {

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

beneficiaryAccounts[i].transfer(inheritance[beneficiaryAccounts[i]]);

}

}

// QUESTION? How would this function get called if the owner is deceased?

function deceased() public onlyOwner {

isDeceased = true;

distributeFunds();

}

}