//SPDX-License-Identifier: MIT

pragma solidity ^0.5.0;

// ROCK, PAPER, SCISSORS

contract RockPaperScissors {

// 2 player game

address payable private player1;

address payable private player2;

string private player1choice;

string private player2choice;

bool private hasPlayer1MadeChoice;

bool private hasPlayer2MadeChoice;

// In order to play the game, the player must pay 1 ether

uint public stake;

// The following is a nested mapping, which contains all possible game results

mapping(string=>mapping(string=>uint8)) private states;

// R - ROCK

// P - PAPER

// S - SCISSORS

// 0 - TIE

// 1 - PLAYER 1 WINS

// 2 - PLAYER 2 WINS

constructor() public {

// initialize the nested mapping

states['R']['R'] = 0;

states['R']['P'] = 2;

states['R']['S'] = 1;

states['P']['R'] = 1;

states['P']['P'] = 0;

states['P']['S'] = 2;

states['S']['R'] = 2;

states['S']['P'] = 1;

states['S']['S'] = 0;

stake = 1 ether;

}

// MODIFIERS

modifier isJoinable() {

require(player1 == address(0) || player2 == address(0), "There are already 2 players!");

require(player1 != address(0) && msg.value == stake || (player1 == address(0)), "You have to pay the stake to play the game!");

\_;

}

modifier isPlayer() {

require(msg.sender == player1 || msg.sender == player2, "You are not a player in the game!");

\_;

}

modifier isValidChoice(string memory \_playerChoice) {

require(keccak256(bytes(\_playerChoice)) == keccak256(bytes('R')) ||

keccak256(bytes(\_playerChoice)) == keccak256(bytes('P')) ||

keccak256(bytes(\_playerChoice)) == keccak256(bytes('S')),

"Choice must be Rock, Paper, or Scissors!");

\_;

}

modifier playerMadeChoice() {

require(hasPlayer1MadeChoice && hasPlayer2MadeChoice, "Both players must make a choice!");

\_;

}

// FUNCTIONS

function join() external payable isJoinable() {

if (player1 == address(0)) {

player1 = msg.sender;

// player1 will determine the stake

stake = msg.value;

} else {

player2 = msg.sender;

}

}

function makeChoice(string calldata \_playerChoice) external isPlayer() isValidChoice(\_playerChoice) {

if(msg.sender == player1 && !hasPlayer1MadeChoice) {

player1choice = \_playerChoice;

hasPlayer1MadeChoice = true;

} else if (msg.sender == player2 && !hasPlayer2MadeChoice) {

player2choice = \_playerChoice;

hasPlayer2MadeChoice = true;

}

}

function showResults() external isPlayer() playerMadeChoice() {

// only the 2 players can reveal the game results, and

// results can only be disclosed after both players have made their choices

int result = states[player1choice][player2choice];

if (result == 0) {

// both players get their 1 ether returned

player1.transfer(stake);

player2.transfer(stake);

} else if (result == 1) {

// player 1 receives 2 ether

player1.transfer(address(this).balance);

} else if (result == 2) {

// player 2 receives 2 ether

player2.transfer(address(this).balance);

}

// Reset the game

player1 = address(0);

player2 = address(0);

player1choice = "";

player2choice = "";

hasPlayer1MadeChoice = false;

hasPlayer2MadeChoice = false;

stake = 1 ether;

}

}