pragma solidity ^0.8.0;

contract StakeOption {

// Define an enumeration for options

enum Option { Option1, Option2, Option3 }

// Mapping to store the total stake for each option

mapping(Option => uint256) public optionStakes;

// Mapping to store the individual stakes for each option

mapping(address => mapping(Option => uint256)) public userStakes;

// Event to let the users know the option with the highest stake

event OptionChosen(Option option);

// Function to let the users place their stake on a specific option

function stake(Option option, uint256 amount) public {

require(amount > 0, "Amount should be greater than 0");

optionStakes[option] += amount;

userStakes[msg.sender][option] += amount;

}

// Function to determine the option with the highest stake and distribute the rewards

function distributeRewards() public {

Option winningOption;

uint256 maxStake = 0;

// Determine the option with the highest stake

for (Option option = Option.Option1; option <= Option.Option3; option++) {

if (optionStakes[option] > maxStake) {

maxStake = optionStakes[option];

winningOption = option;

}

}

// Distribute the rewards to the users who placed their stake on the winning option

for (Option option = Option.Option1; option <= Option.Option3; option++) {

if (option == winningOption) {

continue;

}

for (address user in userStakes) {

uint256 userStake = userStakes[user][option];

if (userStake == 0) {

continue;

}

uint256 reward = userStake \* maxStake / optionStakes[winningOption];

userStakes[user][option] = 0;

userStakes[user][winningOption] += reward;

}

}

// Emit an event to let the users know the winning option

emit OptionChosen(winningOption);

}

}