pragma solidity ^0.5.2;

contract Lottery{

//dynamic array with players payable addresses

//in solidity version newer than 0.5.0 an address must be payable to be able to receive funds (using the transfer function)

address payable[] public players;

address public manager; //contract manager

//contract constructor, runs once at contract deployment

constructor() public{

//the manager is the account address that deploys the contract

manager = msg.sender;

}

//this fallback payable function will be automatically called when somebody

//sends ether to our contract address

//since solidity 0.5.0 it's mandatory to be external

function () payable external{

require(msg.value >= 0.01 ether);

players.push(msg.sender); //add the address of the account that sends

//ether to players array

}

function get\_balance() public view returns(uint){

require(msg.sender == manager);

return address(this).balance; //return contract balance

}

//returns a very big pseodo-random integer no.

function random() public view returns(uint256){

//since solidity 0.5.0 keccak256() function accepts only a single bytes argument

//we use the abi.encodePacked() function to get the bytes argument from 3 values

return uint256(keccak256(abi.encodePacked(block.difficulty, block.timestamp, players.length)));

}

function selectWinner() public {

require(msg.sender == manager);

uint r = random();

address payable winner;

//a random index

uint index = r % players.length;

winner = players[index];

//transfer contract balance to the winner address

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

players = new address payable[](0); //resetting the players dynamic array

}

}