// project

// SPDX-License-Identifier: GPL-3.0

pragma solidity >0.5.0<0.9.0;

/\*

Lottery programme description:

1-manager: manage all things related  to lottery event.(full control)

2- participants: lottery ticket buyers, winner is one.

buy lottery using contract though ether payment.

contract will reset once a round is completed.

\*/

/\* some terms-------------------------------------------------------------------------------

-msg: the adresse who is sending message or calling a function.

-.transfer: to adresse who we ate transfering ether.

-address payable adarsh=payable(..id...); // making adarsh payable .

\*/

// what i did---------------------------------------------------------------------------------:

//initialize  variable(including payable)

//set manager by msg.sender at constructor level

// receive ether if ether value is greather tha threshold 3 :receive() external paybale.

//checking balance :function getbalance

// checking winner address : function winner

// rechecking winner : function selectwinner()

// transferring balance: function transfer()

// restarting new lottery event: function restartLottery()

contract lottery

{

    // store manager address

    address public manager; // to get address of constructor use "msg.sender".

    // adress of winner to whom winned ether will be send

    address payable[] public WinnerParticipants;

    // manager contract adress coding using constructor.

    constructor() // deploy karte hi constructor call hoga.

    {

        // now the caller who is calling this contract, registered as under manager adddress.

        manager=msg.sender; // "msg.sender" is global variable.

    }

    // when an external budy transfer ether, this function will call.

    // function to receive ethers address

    receive() external payable // extranal function is callable only externally.

    // receive is prebuilt function.

    {

        // criteria to be a participant

        //method 2:  required(msg.value==2 ether);

        // now record lottery buyer address.

        require(msg.value >1 ether);

        WinnerParticipants.push(payable(msg.sender));

    }

    function getbalace() view public returns(uint)

    {

        // this pointers parent var

        //.balance to check balance.

        //task: only manager can see the balance

        require(msg.sender==manager); // if caller of function(msg ka sender) possessing id same as manager address.

        return address(this).balance;

    }

    function winner() view public returns(uint)

    // chooseing random winner

    {

    return uint(keccak256(abi.encodePacked(block.difficulty,block.timestamp,WinnerParticipants.length)));

    }

    function selectWinner() public view returns(address)

    {

        // to recheck our winner function

        require(msg.sender==manager);

        require(WinnerParticipants.length>3);

        uint r=winner(); // the winner function we coded, we are genearating  agai a random using same above coded function.

        uint index=r%WinnerParticipants.length; // index must be less than given lenths.

        return WinnerParticipants[index];

    }

    function transfer() public payable returns(address)

    {

        require(msg.sender==manager);

        address payable sriesti=payable(selectWinner());

        sriesti.transfer(getbalace());

        return sriesti;

    }

    function  restartLottery() public  returns(string memory)

    {

        WinnerParticipants=new address payable[](0);

        return "Sriesti lets evote for fut\_tech";

    }

}