```
pragma solidity ^0.4.0;

contract Judge{
    uint public base;
    uint public interval;
    address public owner;

    event isCalled(address _from, uint _time, uint _penalty);

    struct Misbehavior{
        address subject;           //subject who performed the misbehavior;
        address object;            //
        string res;           //
        string action;             //action (e.g., "read", "write", "execute") of the misbehavior
        string misbehavior;        //misbehavior
        uint time;                 //time of the Misbehavior occured
        uint penalty;              //penalty (number of minitues blocked);
    }

    mapping (address => Misbehavior[]) public MisbehaviorList;

    function Judge (uint _base, uint _interval) public{
        base = _base;
        interval = _interval;
        owner = msg.sender;
    }

    function misbehaviorJudge(address _subject, address _object, string _res, string _action, string
_misbehavior, uint _time) public returns (uint penalty){
        //misbehaviorJudge(msb);
        uint length = MisbehaviorList[_subject].length + 1;
        uint n = length/interval;
        penalty = base**n;
        MisbehaviorList[_subject].push(Misbehavior(_subject, _object, _res, _action, _misbehavior, _time,
penalty));
        isCalled(msg.sender,_time, penalty);
    }


    function getLatestMisbehavior(address _key) public constant returns (address _subject, address _object,
string _res, string _action, string _misbehavior, uint _time){
        uint latest = MisbehaviorList[_key].length - 1;
        //uint latest = 0;
        _subject = MisbehaviorList[_key][latest].subject;
        _object = MisbehaviorList[_key][latest].object;
        _res = MisbehaviorList[_key][latest].res;
        _action = MisbehaviorList[_key][latest].action;
        _misbehavior = MisbehaviorList[_key][latest].misbehavior;
        _time = MisbehaviorList[_key][latest].time;
    }

    function self_destruct() public{
        if(msg.sender == owner){
            selfdestruct(this);
        }
    }

```