```
//SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.7.0;

contract Voting {

    struct Voter {
        uint weight;
        bool if_voted;
        address delegated_to;
        uint vote;
    }

    struct Proposal {
        bytes32 name;
        uint voteCount;
    }

    address public chairperson;
    mapping(address => Voter) public voters;
    Proposal[] public proposals;

    constructor(bytes32[] memory proposalNames) public {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;
        for (uint i = 0; i < proposalNames.length; i++) {
            proposals.push(Proposal({name: proposalNames[i], voteCount: 0}));
        }
    }

    function giveRightToVote(address voter) public {
        require(
            msg.sender == chairperson,
            "Only the chairperson can assign voting rights."
        );
        require(!voters[voter].voted, "The voter has used their ballot.");
        require(voters[voter].weight == 0);
        voters[voter].weight = 1;
    }
```

```
function delegate(address to) public {
    Voter storage sender = voters[msg.sender];
    require(!sender.voted, "You have already voted.");
    require(to != msg.sender, "You can’t delegate to yourself.");
    while (voters[to].delegate != address(0)) {
        to = voters[to].delegate;
        require(to != msg.sender, "Found loop in delegation!");
    }
    sender.voted = true;
    sender.delegate = to;
    Voter storage delegate_ = voters[to];
    if (delegate_.voted) {
        proposals[delegate_.vote].voteCount += sender.weight;
    } else {
        delegate_.weight += sender.weight;
    }
}


function vote(uint proposal) public {
    Voter storage sender = voters[msg.sender];
    require(sender.weight != 0, "Cannot vote");
    require(!sender.voted, "Has voted.");
    sender.voted = true;
    sender.vote = proposal;
    proposals[proposal].voteCount += sender.weight;
}



function winningProposal() public view returns (uint winningProposal_) {
    uint winningVoteCount = 0;
    for (uint p = 0; p < proposals.length; p++) {
        if (proposals[p].voteCount > winningVoteCount) {
            winningVoteCount = proposals[p].voteCount;
            winningProposal_ = p;
        }
    }
}


function winnerName() public view returns (bytes32 winnerName_) {
    winnerName_ = proposals[winningProposal()].name;
```

```
    }
}
```