Assignment No.5

Aim:

Polling / voting system using Solidity, Ethereum and a data structure hashmap(optional)

Theory:

- Introduction to smart contract
- Introduction to Solidity programming language

Implementation:

- Write a Smart contract for voting system.
- Connect and deploy smart contract using metamask.
- Design and develop a front end to display the result of election.
- Code:

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.0 <0.9.0;
contract Ballot {
    struct Voter {
        uint weight;
        bool voted;
        address delegate;
        uint vote;
    }

    struct Proposal {
            bytes32 name;
        uint voteCount;
    }

address public chairperson;
```

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

    constructor(bytes32[] memory proposalNames) {
        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) external {
        require(
            msg.sender == chairperson,
            "Only chairperson can give right to vote."
            );
        require(
            !voters[voter].voted,
            "The voter already voted."
        );
        require(voters[voter].weight == 0);
        voters[voter].weight = 1;
    }
    function delegate(address to) external {
        Voter storage sender = voters[msg.sender];
        require(sender.weight != 0, "You have no right to vote");
        require(!sender.voted, "You already voted.");

        require(to != msg.sender, "Self-delegation is disallowed.");

        while (voters[to].delegate != address(0)) {
            to = voters[to].delegate;

            require(to != msg.sender, "Found loop in delegation.");
```

```
        }
        Voter storage delegate_ = voters[to];

        require(delegate_.weight >= 1);

        sender.voted = true;
        sender.delegate = to;

        if (delegate_.voted) {
            proposals[delegate_.vote].voteCount += sender.weight;
        } else {
            delegate_.weight += sender.weight;
        }
    }
    function vote(uint proposal) external {
        Voter storage sender = voters[msg.sender];
        require(sender.weight != 0, "Has no right to vote");
        require(!sender.voted, "Already 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() external view
        returns (bytes32 winnerName_)
```

```
        {
            winnerName_ = proposals[winningProposal()].name;
        }
    }
```

Conclusion:

FAQs:

1. What are dApps and its the benefits?
2. What is the purpose and uses of dApp ?
3. What are the features in dApps?
4. What are the most used dApps?
5. What are the Advantages and Disadvantages of dApps.
6. What is the difference between website and dApp?

Online References:

https://blog.finxter.com/how-does-the-solidity-voting-smart-contract-work/

https://docs.soliditylang.org/en/v0.8.16/solidity-by-example.html