```
contract EVoting{
    address public electionCommission;
    bool public votingStarted;
    bool public votingEnded;
    struct Candidate {
        uint id;
        string name;
        uint voteCount;
    }
    struct Voter {
        bool isRegistered;
        bool hasVoted;
        uint votedCandidateId;
    }
    mapping(uint => Candidate) public candidates;
    mapping(address => Voter) public voters;
    uint public candidatesCount;
    modifier onlyEC() {
        require(msg.sender == electionCommission, "Only Election Commission allowed");
        _;
    }
    constructor() {
        electionCommission = msg.sender;
    }
    // Add candidate before voting starts
    function addCandidate(string memory _name) public onlyEC {
```

```
    require(!votingStarted, "Cannot add candidate after voting starts");
    candidatesCount++;
    candidates[candidatesCount] = Candidate(candidatesCount, _name, 0);
}

// Register a voter before voting starts
function registerVoter(address _voter) public onlyEC {
    require(!votingStarted, "Cannot register after voting starts");
    voters[_voter].isRegistered = true;
}

// Start voting
function startVoting() public onlyEC {
    require(!votingStarted, "Voting already started");
    votingStarted = true;
}

// End voting
function endVoting() public onlyEC {
    require(votingStarted, "Voting not started yet");
    votingEnded = true;
}

// Vote for candidate
function vote(uint _candidateId) public {
    require(votingStarted, "Voting has not started");
```

```
        require(!votingEnded, "Voting has ended");
        require(voters[msg.sender].isRegistered, "Not a registered voter");
        require(!voters[msg.sender].hasVoted, "Already voted");
        require(_candidateId > 0 && _candidateId <= candidatesCount, "Invalid candidate");

        voters[msg.sender].hasVoted = true;
        voters[msg.sender].votedCandidateId = _candidateId;
        candidates[_candidateId].voteCount++;
    }

    // Get results (only after voting ends)
    function getResults() public view returns (string memory winnerName, uint winnerVotes)
{
        require(votingEnded, "Voting is not yet ended");

        uint maxVotes = 0;
        uint winnerId;

        for (uint i = 1; i <= candidatesCount; i++) {
            if (candidates[i].voteCount > maxVotes) {
                maxVotes = candidates[i].voteCount;
                winnerId = i;
            }
        }

        winnerName = candidates[winnerId].name;
```

```
        winnerVotes = candidates[winnerId].voteCount;
    }
}
```