```
pragma solidity >=0.4.22 <0.6.0;

contract EthWill {

    enum State {INITIATED, COMPLETE}
    State currentState;

    address benefactor;
    address representative;
    address payable[] beneficiaries;
    mapping(address => uint) public assigned_eth;

    constructor(address executer) public {
        require(executer != msg.sender);
        benefactor = msg.sender;
        representative = executer;
    }

    /*-----------------<Modifiers>--------------------*/
    modifier onlyBenefactor() {
        require(msg.sender == benefactor);
        _;
    }
    modifier restricted() {
        require(msg.sender == representative || msg.sender == benefactor);
        _;
    }

    modifier inState(State expectedState) {
        require(currentState == expectedState);
        _;
    }

    /*-----------------<Will Functions>--------------------*/
    function assign_eth(address payable[] memory receivers, uint256[] memory eth_finney_amnts) onlyBenefactor pa
yable public {
        require(msg.value >= totalArray(eth_finney_amnts) * 1 finney);
        for(uint i = 0; i < receivers.length; i ++) {
            beneficiaries.push(receivers[i]);
            assigned_eth[receivers[i]] = eth_finney_amnts[i];
        }
        currentState = State.INITIATED;
    }

    function execute_will() public inState(State.INITIATED) restricted {
        for(uint i = 0; i < beneficiaries.length; i ++) {
            beneficiaries[i].transfer(assigned_eth[beneficiaries[i]] * 1 finney);
        }
        currentState = State.COMPLETE;
    }

    function resetWill(uint wei_amount) public inState(State.INITIATED) onlyBenefactor {
        for(uint i = 0; i < beneficiaries.length; i++) {
            assigned_eth[beneficiaries[i]] = 0;
        }
```

```
        delete beneficiaries;
        msg.sender.transfer(wei_amount * 1 wei);
    }

    function changeRepresentative(address rep) public onlyBenefactor {
        representative = rep;
    }

    /*-----------------<Helper>--------------------*/
    function totalArray(uint256[] memory amounts) private pure returns (uint) {
        uint total;
        for(uint i = 0; i < amounts.length; i++) {
            total += amounts[i];
        }
        return total;
    }
}
```