```
// SPDX-License-Identifier: MIT

pragma solidity 0.8.13;

import "@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol";
import "@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol";

contract Precision is Initializable, OwnableUpgradeable {
    uint256 constant public tokensPerEth = 10;
    uint256 constant public weiPerEth = 1e18;
    mapping(address => uint256) public balances;
    address private _owner;

    function initialize () public initializer {
        _owner = msg.sender;
    }

    function mintTokens(address _to, uint256 _amount) public onlyOwner {
        balances[_to] += _amount;
    }

    function buyTokens() public payable {
        uint256 tokens = msg.value * tokensPerEth / weiPerEth; // convert
wei to eth then multiply per token rate
        balances[msg.sender] += tokens;
    }

    function sellTokens(uint256 _tokens) public payable {
        require(balances[msg.sender] >= _tokens);
        uint256 amount = _tokens * weiPerEth / tokensPerEth ;
        balances[msg.sender] -= _tokens;


payable(address(0x000000000000000000000000000000dEAdbEEf0)).transfer(amoun
t); // sending to an inexistent address so that we can easily test ETH
balance before and after calling sellTokens function (if testing on
msg.sender we need to calculate and subtract gas used)
    }

}
```