assignmentToken

Ferdinand S. Ytteborg

October 11, 2021

1 Complete smart contract code

```
/**
 *Submitted for verification at Etherscan.io on 2021-10-11
*/

// SPDX-License-Identifier: MIT
pragma solidity ^0.7.0;

contract assignmentToken {
    // TODO: specify `MAXSUPPLY`, declare `minter` and `supply`
    uint256 constant MAXSUPPLY = 1000000;
    address minter = address(0);
    uint256 supply = 50000;

    // TODO: specify event to be emitted on transfer
    event Transfer(address indexed _from, address indexed _to, uint256 _value);

    // TODO: specify event to be emitted on approval
    event Approval(address indexed _owner, address indexed _spender, uint256 _value
        );

    event MintershipTransfer(
        address indexed previousMinter,
        address indexed newMinter
    );

    // TODO: create mapping for balances
    mapping (address => uint) public balances;

    // TODO: create mapping for allowances
    mapping (address => mapping(address => uint)) public allowances;

    constructor() {
        // TODO: set sender's balance to total supply
        balances[msg.sender] = supply;
        minter = msg.sender;
    }

    function totalSupply() public view returns (uint256) {
        // TODO: return total supply
        return supply;
    }

    function balanceOf(address _owner) public view returns (uint256) {
        // TODO: return the balance of _owner
        return balances[_owner];
    }

    function mint(address receiver, uint256 amount) public returns (bool) {
        // TODO: mint tokens by updating receiver's balance and total supply
        // NOTE: total supply must not exceed `MAXSUPPLY`
        require(supply + amount <= MAXSUPPLY);
        require(msg.sender == minter);
        supply += amount;
        balances[receiver] += amount;
        return true;
    }

    function burn(uint256 amount) public returns (bool) {
        // TODO: burn tokens by sending tokens to `address(0)`
        // NOTE: must have enough balance to burn
```

```
60          require(balances[msg.sender] >= amount);
61          send(msg.sender, address(0), amount);
62          supply -= amount;
63          emit Transfer(msg.sender, address(0), amount);
64          return true;
65      }
66
67      function transferMintership(address newMinter) public returns (bool) {
68          // TODO: transfer mintership to newminter
69          // NOTE: only incumbent minter can transfer mintership
70          // NOTE: should emit `MintershipTransfer` event
71          require(msg.sender == minter);
72          minter = newMinter;
73          emit MintershipTransfer(msg.sender, newMinter);
74          return true;
75      }
76
77      function send(address _from, address _to, uint256 _value) private{
78          balances[_from] -= _value;
79          balances[_to] += _value;
80      }
81
82      function transfer(address _to, uint256 _value) public returns (bool) {
83          // TODO: transfer `_value` tokens from sender to `_to`
84          // NOTE: sender needs to have enough tokens
85          // NOTE: transfer value needs to be sufficient to cover fee
86          require(balances[msg.sender] >= _value + 1);
87          send(msg.sender, _to, _value);
88          send(msg.sender, minter, 1); //transaction fee
89          emit Transfer(msg.sender, _to, _value);
90          return true;
91      }
92
93      function transferFrom(
94          address _from,
95          address _to,
96          uint256 _value
97      ) public returns (bool) {
98          // TODO: transfer `_value` tokens from `_from` to `_to`
99          // NOTE: `_from` needs to have enough tokens and to have allowed sender to
                spend on his behalf
100         // NOTE: transfer value needs to be sufficient to cover fee
101         require(balances[_from] >= _value + 1, "balances too low");
102         require(allowances[_from][msg.sender] >= _value + 1, "allowances too low");
103         allowances[_from][msg.sender] -= _value + 1;
104         send(_from, _to, _value);
105         send(_from, minter, 1);
106         emit Transfer(_from, _to, _value);
107         return true;
108     }
109
110     function approve(address _spender, uint256 _value) public returns (bool) {
111         // TODO: allow `_spender` to spend `_value` on sender's behalf
112         // NOTE: if an allowance already exists, it should be overwritten
113         allowances[msg.sender][_spender] = _value;
114         emit Approval(msg.sender, _spender, _value);
115         return true;
116     }
117
118     function allowance(address _owner, address _spender)
119         public
120         view
121         returns (uint256 remaining)
122     {
123         // TODO: return how much `_spender` is allowed to spend on behalf of `
                _owner`
124         remaining = allowances[_owner][_spender];
125         return remaining;
126     }
127 }
```

2 Deployed smart contract url

https://kovan.etherscan.io/address/0x06b1af890b29dbb4d7a50911dc0eabcafaf3522f

3 Transaction urls

1. https://kovan.etherscan.io/tx/0x14c7f2c349b4b0287a7e03cc593cfb7398042cd71e268e2345bc2652241b6164
2. https://kovan.etherscan.io/tx/0x95e8972bf8e6f3aa8b5994f01ef28a932d87f4ee2ead52197f2d39b3e6ec6337
3. https://kovan.etherscan.io/tx/0x5ba361f546c9344a1af3b3f9ff9fe6768076b0affd8bc6981d0f3d51fff44ab3
4. https://kovan.etherscan.io/tx/0x3184e9491e96fca865e1151d03fa40a3d5a355fcf51de6a7b5ef4a1711a8f381
5. https://kovan.etherscan.io/tx/0xe688d50c6e394a042b66b4774d4045d542eb46b25f1053cb1ab99ffe85daada1