Student ID: 21180859

Smart Contract Code:

```
pragma solidity ^0.7.0;

contract assignmentToken {

    uint256 supply = 50000; // Setting the initial supply.

    uint256 constant MAXSUPPLY = 1000000; // Specifying the maximum supply.
    uint256 constant fee = 1; // Specifying the fee.

    address public minter; // Setting the minter to be a public variable
(visible to the blockchain)

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

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

     // event to be emitted on mintership transfer
    event MintershipTransfer(address indexed previousMinter, address indexed
newMinter);

    // mapping for balances
    mapping(address => uint256) public balances;

    // mapping for allowances
    mapping(address => mapping(address => uint256)) public allowances;

    constructor() {

        balances[msg.sender] = supply; // Setting the contract creator to have
balance equal to supply
        minter = msg.sender;  // Setting the original minter to be the
contract creator.

    }

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

```solidity
    function balanceOf(address _owner) public view returns (uint256) {

        return balances[_owner];
    }

    function mint(address receiver, uint256 amount) public returns (bool) {
        // mint tokens by updating receiver's balance and total supply
        require((supply + amount) <= MAXSUPPLY);
        require(msg.sender == minter);

        balances[receiver] += amount;
        supply += amount;
        return true;
    }

    function burn(uint256 amount) public returns (bool) {
        // burn tokens by sending tokens to `address(0)`
        require(amount <= balances[msg.sender]);
        balances[msg.sender] -= amount;
        supply -= amount;

        emit Transfer(msg.sender, address(0), amount);
        return true;
    }

    function transferMintership(address newMinter) public returns (bool) {
        // transfer mintership to newminter
        require(msg.sender == minter);

        minter = newMinter;
        emit MintershipTransfer(msg.sender, newMinter);
        return true;
    }

    function transfer(address _to, uint256 _value) public returns (bool) {
         // transfer `_value` tokens from sender to `_to`
        require(_value <= balances[msg.sender]);
        require(fee <= _value);

        balances[msg.sender] -= _value;
        balances[_to] += _value - fee;
        balances[minter] += fee;
        emit Transfer(msg.sender, _to, _value - fee); // Since this is a log
statement, we're interested in printing how much was transfered.
        return true;
    }
    function transferFrom(address _from, address _to, uint256 _value) public
returns (bool) {
        // TODO: transfer `_value` tokens from `_from` to `_to`
        require(_value <= balances[_from]);
        require(_value <= allowances[_from][msg.sender]);
        require(fee <= _value);

        balances[_from] -= _value;
        allowances[_from][msg.sender] -= _value;
        balances[_to] += _value - fee;
        balances[minter] += fee;

        emit Transfer(_from, _to, _value - fee);
        return true;
    }

    function approve(address _spender, uint256 _value) public returns (bool) {
        //  allow `_spender` to spend `_value` on sender's behalf
        allowances[msg.sender][_spender] = _value;
        emit Approval(msg.sender, _spender, _value);
        return true;
    }

    function allowance(address _owner, address _spender)
        public
        view
        returns (uint256 remaining)
    {
        // return how much `_spender` is allowed to spend on behalf of
`_owner`
        return allowances[_owner][_spender];
    }
}
```

Deployed Smart Contract URL:

https://kovan.etherscan.io/address/0x80666b1ff089acc4a4fe4385e821da56752415ac#code

Transaction urls:

1) **Mint 60 new tokens to an address.**

Minter address: 0x46717Abc4a2c3cf22A06267712a9A932Daf03cb7 ------ ACC 1

Address getting funds: 0x168AA2502E6F2469761557797cc1D3Eb5E88fdC5 ------ ACC 2

url:
https://kovan.etherscan.io/tx/0x35621966852d1451c9122b8f6abdfb25f45e74889daaa813bb20e04
35cdbd0bf

2) **Burn 70 tokens from ACC 1.**

url:
https://kovan.etherscan.io/tx/0x33be25d41b7cfdc5911099ed86a4d1e6626ff62df21a6943084d8eb0
0a73957e

3) **Approve ACC 2 to spend up to 110 tokens from ACC** 1.

url:

https://kovan.etherscan.io/tx/0xb5d685f17be66ff0bc822c6f6d64c52eac445e2f7876acdb461d7d36d
0e126ca

4) **Transfer mintership to ACC 2**

url:

https://kovan.etherscan.io/tx/0x2b1fe39bb8b97bb81751bcb41b4b54a7e8dd7177cb68f898641b2d6
9d59e41a7

5) **Transfer 40 tokens with ACC 2 from ACC 1 to ACC 3**

0xba00dD6df1007fa585557FB4Cda51741804B1bC9 ------ ACC 3

url:

https://kovan.etherscan.io/tx/0x48d65e4a1d23d3bdd822cba8a54734ee097d8b17f5ccc73d6f7f21da
e7ac1370