AI_Blockchain Project A

Code

Git

```
pragma solidity >=0.4.22 <0.7.0;

contract ERC20Basic {

    string public constant name = "R07521606 Token";
    string public constant symbol = "R07521606";
    uint8 public constant decimals = 18;


    event Approval(address indexed tokenOwner, address indexed spender, uint
tokens);
    event Transfer(address indexed from, address indexed to, uint tokens);


    mapping(address => uint256) balances;

    mapping(address => mapping (address => uint256)) allowed;

    uint256 _totalSupply;

    using SafeMath for uint256;

    // constructor to define totalSupply of this erc20 token
    constructor(uint256 total) public {
      _totalSupply = total;
      balances[msg.sender] = _totalSupply;
    }

    // get totalSupply
    function totalSupply() public view returns (uint256) {
      return _totalSupply;
    }

    // get balance of a specific account
```

```
    function balanceOf(address tokenOwner) public view returns (uint) {
        return balances[tokenOwner];
    }

    // transfer tokens from msg.sender to a receiver
    function transfer(address receiver, uint tokensNum) public returns (bool) {
        // error handler: ensure that the balance is more than the number of tokens
to be sent
        require(tokensNum <= balances[msg.sender]);
        balances[msg.sender] = balances[msg.sender].sub(tokensNum);
        balances[receiver] = balances[receiver].add(tokensNum);
        emit Transfer(msg.sender, receiver, tokensNum);
        return true;
    }

    // to allow an owner i.e. msg.sender to approve a delegate account to
    // withdraw tokens from his account and to transfer them to other accounts
    function approve(address delegate, uint numTokens) public returns (bool) {
        allowed[msg.sender][delegate] = numTokens;
        emit Approval(msg.sender, delegate, numTokens);
        return true;
    }

    // get count of allowance tokens
    function allowance(address owner, address delegate) public view returns (uint) {
        return allowed[owner][delegate];
    }

    // transfer by delegate
    function transferFrom(address owner, address buyer, uint numTokens) public
returns (bool) {
        // error handler: ensure that the balance is more than the number of tokens
to be sent
        require(numTokens <= balances[owner]);
        // error handler: ensure that the allowance is more than the number of
tokens to be sent
        require(numTokens <= allowed[owner][msg.sender]);

        balances[owner] = balances[owner].sub(numTokens);
        allowed[owner][msg.sender] = allowed[owner][msg.sender].sub(numTokens);
        balances[buyer] = balances[buyer].add(numTokens);
        emit Transfer(owner, buyer, numTokens);
        return true;
    }
```

```
}

// SafeMath is a Solidity library aimed at dealing with one way hackers have been known to break contracts: integer overflow attack
library SafeMath {
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
      assert(b <= a);
      return a - b;
    }

    function add(uint256 a, uint256 b) internal pure returns (uint256) {
      uint256 c = a + b;
      assert(c >= a);
      return c;
    }
}
```

Address of Contract

> 0x6a75b5e7b45ead66cab062b33c77333a8986c9ca

Screenshot of Transaction on Etherscan