```
pragma solidity ^0.4.8;


contract SafeMath {
    function assert(bool assertion) internal {
        if (!assertion) {
            throw;
        }
     }


    function safeAdd(uint256 x, uint256 y) internal returns(uint256) {
      uint256 z = x + y;
      assert((z >= x) && (z >= y));
      return z;
    }


    function safeSubtract(uint256 x, uint256 y) internal returns(uint256) {
      assert(x >= y);
      uint256 z = x - y;
      return z;
    }


    function safeMult(uint256 x, uint256 y) internal returns(uint256) {
        uint256 z = x * y;

        assert((x == 0)||(z/x == y));

        return z;

    }


}


contract Token {

    uint256 public totalSupply;

    function balanceOf(address _owner) constant returns (uint256 balance);

    function transfer(address _to, uint256 _value) returns (bool success);

    function transferFrom(address _from, address _to, uint256 _value) returns
(bool success);

    function approve(address _spender, uint256 _value) returns (bool success);

    function allowance(address _owner, address _spender) constant returns
(uint256 remaining);

    event Transfer(address indexed _from, address indexed _to, uint256 _value);

    event Approval(address indexed _owner, address indexed _spender, uint256
_value);

}
```

```
/*   ERC 20 token */
contract StandardToken is Token {

    function transfer(address _to, uint256 _value) returns (bool success) {
      if (balances[msg.sender] >= _value && _value > 0) {
        balances[msg.sender] -= _value;
        balances[_to] += _value;
        Transfer(msg.sender, _to, _value);
        return true;
      } else {
        return false;
      }
    }

    function transferFrom(address _from, address _to, uint256 _value) returns
(bool success) {
      if (balances[_from] >= _value && allowed[_from][msg.sender] >= _value
&& _value > 0) {
        balances[_to] += _value;
        balances[_from] -= _value;
        allowed[_from][msg.sender] -= _value;
        Transfer(_from, _to, _value);
```

```
            return true;
        } else {
            return false;
        }
    }

    function balanceOf(address _owner) constant returns (uint256 balance) {
        return balances[_owner];
    }

    function approve(address _spender, uint256 _value) returns (bool success) {
        allowed[msg.sender][_spender] = _value;
        Approval(msg.sender, _spender, _value);
        return true;
    }

    function allowance(address _owner, address _spender) constant returns
(uint256 remaining) {
      return allowed[_owner][_spender];
    }

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

contract ACToken is StandardToken, SafeMath {

    string public constant name = "AC Token";
    string public constant symbol = "ACT";
    uint256 public constant decimals = 18;
    string public version = "1.0";

    address public ethFundDeposit;
    address public actFundDeposit;

    bool public isFinalized;
    bool public isHalted;
    uint256 public fundingStartBlock;
    uint256 public fundingEndBlock;
    uint256 public constant factorial = 6;
    uint256 public constant actFund = 49 * (10**factorial) * 10**decimals; //49%
锁定代币地址，共计 49M 即 4900 万代币
    uint256 public constant tokenCreationCap = 100 * (10**factorial) *
10**decimals; /// 最大募集金额 51M,即 5100 万代币
```

```
}

    uint256 public constant p0Rate = 125000;

    uint256 public constant p1Rate = 115000;

    uint256 public constant p2Rate = 108000;

    uint256 public constant p3Rate = 100000;


    uint256 public constant p0Period = 2000; //P0 时间长度 1 小时，P0 比例
12500， +25%

    uint256 public constant p1Period = 10000; //P1 时间长度大约 1 天，P1 比例
11500， +15%

    uint256 public constant p2Period = 50000; //P2 时间长度大约 9 天，P2 比例
10800， +8%


    function tokenRate() constant returns(uint) {

        if (block.number>=fundingStartBlock &&
block.number<fundingStartBlock+p0Period) return p0Rate;

        if (block.number>=fundingStartBlock &&
block.number<fundingStartBlock+p1Period) return p1Rate;

        if (block.number>=fundingStartBlock &&
block.number<fundingStartBlock+p2Period) return p2Rate; // first week

        return p3Rate;

    }
```

```
// events
event CreateACT(address indexed _to, uint256 _value);


// constructor
function ACToken(
        address _ethFundDeposit,
        address _actFundDeposit,
        uint256 _fundingStartBlock,
        uint256 _fundingEndBlock)
{
    isFinalized = false;
    isHalted = false;
    ethFundDeposit = _ethFundDeposit;
    actFundDeposit = _actFundDeposit;

    fundingStartBlock = _fundingStartBlock;
    fundingEndBlock = _fundingEndBlock;

    totalSupply = actFund;
    balances[actFundDeposit] = actFund;
    CreateACT(actFundDeposit, actFund);
}



function makeTokens() payable   {
    if (isFinalized) throw;
    if (isHalted) throw;
    if (block.number < fundingStartBlock) throw;
    if (block.number > fundingEndBlock) throw;
    if (msg.value == 0) throw;

    uint256 tokens = safeMult(msg.value, tokenRate());

    uint256 checkedSupply = safeAdd(totalSupply, tokens);

    if (tokenCreationCap < checkedSupply) throw;

    totalSupply = checkedSupply;
    balances[msg.sender] += tokens;
    CreateACT(msg.sender, tokens);
}
```

```
        function() payable {
                makeTokens();
        }

        function finalize() external {
            if (isFinalized) throw;
            if (msg.sender != ethFundDeposit) throw;

            if(block.number <= fundingEndBlock && totalSupply != tokenCreationCap)
    throw;

            isFinalized = true;
            if(!ethFundDeposit.send(this.balance)) throw;
        }

        function sendEth(uint amount) {
            if (msg.sender != ethFundDeposit) throw;
            uint ethAmount = amount * 1 finney;
            if(!ethFundDeposit.send(ethAmount)) throw;
        }

        function halt() {
```

```
        if (msg.sender != ethFundDeposit) throw;

        isHalted = true;

    }


    function unhalt() {

        if (msg.sender != ethFundDeposit) throw;

        isHalted = false;

    }

}
```