

作业

```
wangyang — root@docker-desktop: /opt/sandbox/alg — lisp.run -B /usr/local/Cellar/clisp/2.49_2/lib/clisp-2.49 -M /usr/local/Cellar/...
Emacs Lisp files have been installed to:
/usr/local/share/emacs/site-lisp/clisp
[YangdeMBP:~ wangyang$ clisp
i i i i i i      00000      0      0000000      00000      00000
I I I I I I      8      8      8      8      0      8      8
I \ \ '+ ' / I    8      8      8      8      8      8
  \ - '+ - /      8      8      8      00000      80000
   \ - '+ - /      8      8      8      8      8
    |              8      0      8      8      0      8      8
-----+-----    00000      8000000      0008000      00000      8

Welcome to GNU CLISP 2.49 (2010-07-07) <http://clisp.cons.org/>

Copyright (c) Bruno Haible, Michael Stoll 1992, 1993
Copyright (c) Bruno Haible, Marcus Daniels 1994-1997
Copyright (c) Bruno Haible, Pierpaolo Bernardi, Sam Steingold 1998
Copyright (c) Bruno Haible, Sam Steingold 1999-2000
Copyright (c) Sam Steingold, Bruno Haible 2001-2010

Type :h and hit Enter for context help.

[[1]> (+ 1 2)
3
[2]> ]
```

```
[root@d83ade9ee1c1:/src/blockstack-core# clarity-cli generate_address
SP2QACC2YCX51G0K42R7KE1PWD3REDW9W4D35RQT5
```

```
root@d83ade9ee1c1:/src/blockstack-core# export DEMO_ADDRESS=SP2QACC2YCX51G0K42R7KE1PWD3REDW9W4D35RQT5
```

```
root@d83ade9ee1c1:/src/blockstack-core# echo $DEMO_ADDRESS
SP2QACC2YCX51G0K42R7KE1PWD3REDW9W4D35RQT5
```

```
[root@d83ade9ee1c1:/src/blockstack-core/sample-programs# clarity-cli execute /data/db/ $DEMO_ADDRESS.sum sum $DEMO_ADDRESS u10 u215
Transaction executed and committed. Returned: u225
```

```

1 (define-map tokens ((account principal)) ((balance uint))) ;定义名为tokens
  的map, key为持有地址, value为token数量
2 (define-private (get-balance (account principal)) ;定义私有函数, 入参为用户
  地址
3 (default-to u0 (get balance (map-get? tokens (tuple (account
  account)))))) ;返回输入地址持有的token数量
4
5 (define-private (token-credit! (account principal) (amount uint));定义私有
  函数token-credit
6 (if (<= amount u0) ;如果amount<=0
7 (err "must move positive balance") ;抛出错误, 请输入正数
8 (let ((current-amount (get-balance account))) ;定义current-ammount=输入地
  址的当前token数量
9 (begin
10 (map-set tokens (tuple (account account))

```

```

11  (tuple (balance (+ amount current-amount)))) ;给tokens中的的入参地址的
    key+amount的token
12  (ok amount))))))
13
14  (define-public (token-transfer (to principal) (amount uint));定义一份公有
    方法, token-transfer, 参数为收款人地址和转账数
15  (let ((balance (get-balance tx-sender))) ;定义变量balance=调用发起者的tok
    en数量
16  (if (or (> amount balance) (<= amount u0)) ;如果转账数量超过持有token或转
    账数额<=0则抛出错误
17  (err "must transfer positive balance and possess funds")
18  (begin
19  (map-set tokens (tuple (account tx-sender))
20  (tuple (balance (- balance amount)))) ;给转账发起者减去amount数量的token
21  (token-credit! to amount)))) ;给接受者增加token
22
23  (define-public (mint! (amount uint)) ;定义函数mint!, 入参为amount
24  (let ((balance (get-balance tx-sender))) ;定义变量balance=调用者持有toke
    n数量
25  (token-credit! tx-sender amount))) ;给调用者增加amount的token
26
27  (token-credit! 'SZ2J6ZY48GV1EZ5V2V5RB9MP66SW86PYKKQ9H6DPR u10000) //给地
    址SZ2J..6DPR增加10000token
28  (token-credit! 'SM2J6ZY48GV1EZ5V2V5RB9MP66SW86PYKKQVX8X0G u300) //给地址S
    Z2J..8X0G增加300token
29

```

思考题目

题目1：

智能合约带来的是逻辑代码的公开。一方面带来了开放，另一方面也带来了安全隐患，对编码的安全性要求大大提高了

题目2：

可以将一些原本放在radiks内的重要数据，比如dapp积分等放到智能合约中，其他的一些不涉及隐私or安全的公共值依旧可以放在radiks，更多的个人隐私数据依旧放在gaia中，不冲突

题目3：

留言板中可以不改动。也可以将原本存在radiks中的数据存放到合约的map中
publicMessage为明文数据，其他私有group存储加密数据

```
define-map message ((publicMessage buffer) (group1 buffer) ...) ((messageList list)
(messageList list) ...)
```