作业

- 搭建Clarity环境，跑通简单的智能合约并将过程截图提交到screenshot文件夹中

```
root@c5956cc79cc0:/src/blockstack-core/sample-programs# cat demo.clar
(define-public (demo)
  (ok "just a demo"))

root@c5956cc79cc0:/src/blockstack-core/sample-programs# clarity-cli check
demo.clar
Checks passed.
root@c5956cc79cc0:/src/blockstack-core/sample-programs# clarity-cli launch
$ADDRESS.demo demo.clar /data/db
Contract initialized!
root@c5956cc79cc0:/src/blockstack-core/sample-programs# clarity-cli execute
/data/db $ADDRESS.demo demo $ADDRESS
Transaction executed and committed. Returned: 0x6a75737420612064656d6f
```

- 分析token.clar代码，将带有注释的token.clar代码提交到screenshot文件夹中

```
(define-map tokens ((account principal)) ((balance uint)))  ;; 定义名为tokens的
map结构键值对,指定键值对名和类型

(define-private (get-balance (account principal)) ;; 定义私有函数,获取balance
  (default-to u0 (get balance (map-get? tokens (tuple (account account))))))
;; 自定义tuple存储账户,获取tokens的balance,默认为0

(define-private (token-credit! (account principal) (amount uint)) ;; 私有函数
token-credit,接收参数为账户名和amount
  (if (<= amount u0) ;; 检查amount是否正常,小于0则报错,否则调用私有函数获取balance
      (err "must move positive balance")
      (let ((current-amount (get-balance account))) ;; 修改balance数量
        (begin
          (map-set tokens (tuple (account account))
                   (tuple (balance (+ amount current-amount)))) ;; balance=
amount+从get-balance方法获取到的账户balance
          (ok amount))))) ;; 表示成功则返回amount

(define-public (token-transfer (to principal) (amount uint)) ;; 公有方法,用于转出
amount
  (let ((balance (get-balance tx-sender))) ;; 临时获取balance
    (if (or (> amount balance) (<= amount u0)) ;; 检查balance是否正常
```

```
        (err "must transfer positive balance and possess funds") ;; 异常则报错
        (begin   ;; 开始执行转移amount逻辑
          (map-set tokens (tuple (account tx-sender)) ;; 使用一开始定义的tokens键值对结构
                       (tuple (balance (- balance amount)))) ;; 设置新的balance=balance-amount
          (token-credit! to amount))))) ;;调用私有函数,接收方即可获取到amount

(define-public (mint! (amount uint)) ;; 共有函数mint
   (let ((balance (get-balance tx-sender))) ;; 查tx-sender balance
     (token-credit! tx-sender amount))) ;; 转出amount到tx-sender

(token-credit! 'SZ2J6ZY48GV1EZ5V2V5RB9MP66SW86PYKKQ9H6DPR u10000) ;; 调用token-credit函数给指定的地址转10000

(token-credit! 'SM2J6ZY48GV1EZ5V2V5RB9MP66SW86PYKKQVX8X0G u300) ;; 转300
```

- 回答思考题，将思考题答案提交到screenshot文件夹中
  - 题目一：根据今天对于智能合约的讲解，你认为智能合约可以解决哪些现有互联网无法解决的问题？又会带来哪些问题？
    - 粗浅的理解,应该能解决互联网中第三方信任问题,使用智能合约就不再需要第三方进行介入.
  - 题目二：前六节课的主要内容均为Blockstack V1的架构，本节课为Blockstack V2架构中的一个核心内容，请问你认为V1与V2将如何结合在一起呢？
    - 这不是我应该考虑的问题,我不想扯淡.
  - 题目三：如果将本节课的内容应用在去中心化留言板中，你认为整个留言板的流程图会有什么变化？会添加哪些功能？（建议画出流程图讲解）
    - 不清楚.也许能主动给其他人传递信息.