Homework 3 - Data Abstractions

本次作业需要在 Online Judge 上提交（校内网），校外可以使用学校VPN。服务器由于断电、重启等原因也可能短暂地不可用，请同学们在本地留存好自己的代码。

默认每题时限为 180 秒，内存限制 1024 MB，栈大小同内存限制，正确实现的代码可以以远低于该时空限制的要求下通过所有测试用例。提交答案需要完整地赋值黏贴对应题目的源码，不包括测试代码。每题满分均为 100，你的得分为 $\left\lceil \frac{\#\text{passed}}{\#\text{total}} \right\rceil$，其中 #passed 为通过的测试用例数量，#total 为该题的测试用例数量，以最后一次提交的得分为准。作业截止时间以 OJ 为准。

遇到问题可以联系助教 23110240130@m.fudan.edu.cn。

Note

为了更方便地实现数据抽象，我们需要学习一些课上没讲过的东西，这能更好地帮助你完成将来的解释器大作业，本次作业我们将练习使用 Racket 中的结构体 `struct`。同时 Racket 也允许我们像 Haskell 或 OCaml 一样使用模式匹配 `match` 来访问数据，速通教程如下（`example.rkt`）。

```racket
#lang racket

(struct Point (x y)) ; 声明一个结构体类型 Point，有两个域 x 和 y

; 这相当于你自动获得了一个 constructor、一个 predicate 和若干 accessor
(define p_a (Point 1 2))       ; constructor
(println (Point? p_a))         ; #t
(println (Point? (cons 1 2))) ; #f
(println (Point-x p_a))        ; 1
(println (Point-y p_a))        ; 2

(struct Line (p1 p2))
(struct Circle (p r))
(define line_1 (Line p_a (Point 3 4)))
(define circle_1 (Circle p_a 2))

; 使用模式匹配访问数据
; (match val [cond1 e1] [cond2 e2] ...) 的语义是：
; 1. 若 val 满足模式 cond1 则整个表达式值为 e1
; 2. 若 val 满足模式 cond2 则整个表达式值为 e2
; ...
; 匹配是从上往下顺序进行的，模式可以嵌套，下划线表示通配符
; (? pred pat ...) 表示当 (pred val) 为 #t 且 val 满足模式 pat 时才匹配
;
; 注：[] 和 () 是可以替换的，这里混合使用只是为了看得清楚
(define (my-print s)
  (match s
    [(Point 0 0) (printf "origin\n")]
```

```
29      [(Point 2 _) (printf "a point with x=2\n")]
30      [(Point _ _) (printf "a point\n")]
31      [7 (printf "a lucky number 7\n")]
32      [(? number? x) (printf "just a number ~a\n" x)]
33      [(Line (Point x1 y1) (Point x2 y2))
34       (printf "a line from (~a,~a) to (~a,~a)\n" x1 y1 x2 y2)]
35      [(Circle (Point x y) r)
36       (printf "a circle with a radius of ~a and a center at (~a,~a)\n" r x y)]
37      [_ (printf "i don't know how to print\n")]))
38
39  (my-print 12)                ; just a number 12
40  (my-print 7)                 ; a lucky number 7
41  (my-print (Point 0 0))       ; origin
42  (my-print (Point 2 3))       ; a point with x=2
43  (my-print (Line-p2 line_1)) ; a point
44  (my-print line_1)            ; a line from (1,2) to (3,4)
45  (my-print circle_1)          ; a circle with a radius of 2 and a center at (1,2)
46  (my-print (list 1 2 3))      ; i don't know how to print
```

进阶用法可参考 The Racket Guide：Programmer-Defined Datatypes & Pattern Matching。

A oddprint

实现函数 `oddprint`，该函数需要接受一个 `list` 并按以下要求转换成字符串：

1. 空列表 → `"empty"`；
2. 长度为 1 且唯一项为 7 → `"one-lucky"`；
3. 长度为 1 → `"one"`；
4. 长度为 2 且第二项是一个 `list` 且第二项的 `list` 的第一项是 7 → `"two-lucky"`；
5. 长度为 2 → `"two"`；
6. 长度为 3 且至少有一项为 4（不考虑递归）→ `"three-unlucky"`；
7. 其他情况 → `"oh"`。

若同时满足多个条件，按编号最小的规则进行转换。

B password

实现函数 `validate`，`(validate pw rule)` 检验字符串 `pw` 是否满足规则 `rule`，规则如下：

1. `(MinimumLength len)` 要求字符串长度大于等于 `len`；
2. `(ContainsSome chars)` 要求字符串至少包含 `chars` 中的某一个字符；

3. `(DoesNotContain chars)` 要求字符串不能包含 `chars` 中的任一个字符；
4. `(And r1 r2)` 要求同时满足规则 `r1` 和 `r2`；
5. `(Or r1 r2)` 要求至少满足规则 `r1` 或 `r2` 中的一个。

C calculator

实现一个仅包含加减乘和取相反数的语言的解释器 `calculate`，该语言的表达式有两种：

1. `u-expr` 为一元运算符，仅包含 `'Neg` 取相反数；
2. `b-expr` 为二元运算符，仅包含 `'Add 'Sub 'Mul` 分别为加减乘。