Homework 1 - Higher-Order Procedures

本次及以后作业我们尝试采用 Online Judge（校内网），校外可以使用学校VPN。账号和初始密码均为学号，请登录后立即修改自己的密码。遇到问题可以联系助教 23110240130@m.fudan.edu.cn。

默认每题时限为 180 秒，内存限制 1024 MB，栈大小同内存限制。提交答案需要完整地赋值黏贴对应题目的源码。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，请大家不要过于频繁地提交代码，通过本地测试后再提交。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。

由于我们是初次尝试使用 OJ 进行评测，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，使用中遇到任何问题也可以联系助教。

OJ 上的 Homework 0 仅为功能测试，请大家忽略。

Note

所谓高阶函数就是以函数为参数或者以函数为返回值的函数。你需要实现五个非常常见的高阶函数，请不要通过 Racket 自带的高阶函数来实现。

每题都有测试代码，例如可以通过执行 `racket curry-test.rkt` 对 `curry` 这题进行测试。

A curry

Curring 也称柯里化，参考 wikipedia。

`curry` 函数的类型为 $((a, b) \to c) \to a \to b \to c$，含义为它可以把一个 “接受 a 类型和 b 类型参数并返回 c 类型的函数” 变成一个 “接受 a 类型的参数并返回 ‘接受 b 类型的参数并返回 c 类型的函数’ 的函数”。

这个套娃的实际意义在于告诉我们 $(a, b) \to c$ 和 $a \to b \to c$ 没有本质区别，即所有多参数的函数都可以被转换为单个参数的函数的组合。

你可以通过阅读 `curry-test.rkt` 来了解用法。

B compose

`compose` 为函数复合，类型为 $(b \to c, a \to b) \to (a \to c)$，`(compose f g)` 的含义为 $f \circ g$。

C map

`map` 函数的类型为 $(a \to b, [a]) \to [b]$，含义为接受一个把 a 类型数据映射到 b 类型数据的函数，并把该函数逐个应用于列表中的每个元素，得到一个 b 类型构成的列表。

你还需要利用 `map` 函数实现 `get-col`。`(get-col x mat)` 的含义为求矩阵 `mat` 中的第 `x` 列的转置，其中保证 `mat` 是一个长度相同的列表构成的列表， `x` 大于等于 1 且小于等于矩阵列数。

例如 `(get-col 2 '((1 2 3) (4 5 6) (7 8 9)))` 应该得到 `'(2 5 8)`。

D filter

`filter` 函数的类型为 $(a \to \text{Bool}, [a]) \to [a]$，意思是接受一个把 a 类型数据映射到布尔类型数据的函数，并把使该函数为真的那些元素保留下来。

你还需要利用 `filter` 函数实现 `intersect`。`(intersect lst1 lst2)` 将产生一个新列表，新列表中的每个元素是 `lst1` 和 `lst2` 的交集，并且在列表中的顺序与在 `lst1` 相同。

例如 `(intersect '(1 2 3) '(2 3 4))` 应该得到 `'(2 3)`。

E foldr

`foldr` 函数的类型为 $((a, b) \to b, b, [a]) \to b$ ，含义为从右往左把列表变成折叠成一个值。

比如 `(foldr + 0 (list x y z))` 求的就是 $x + (y + (z + 0))$。

你还需要利用 `foldr` 函数实现 `group-adjacent`。`(group-adjacent xs)` 会把列表 `xs` 中相邻的相同元素分组并返回分组列表，分组应保持元素的原始顺序。

例如 `(group-adjacent '(1 1 2 2 2 3 1))` 应该得到 `'((1 1) (2 2 2) (3) (1))`。