Homework 2 - More High-Order Procedures

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。和 HW1 不同的是，这次我们会在 OJ 上加入隐藏的测试用例，本地测试通过不代表 OJ 上可以获得满分，请大家充分测试后再提交。

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

Note

写解释器不可避免地要处理抽象语法树，因此我们来搞树！

作业中你可能会使用到 `null?`、`car`、`cdr` 等列表相关操作，以及 `foldr`、`map` 等高阶函数。

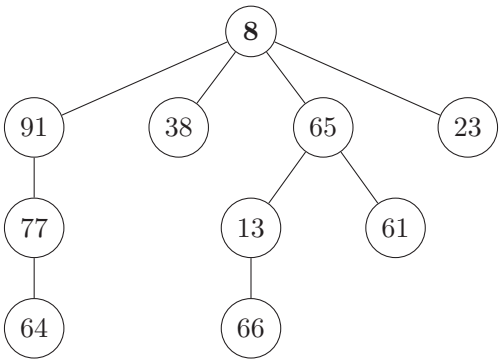
A tree

A.1 fold-tree

实现函数 `(fold-tree tree f)`，它用来深度优先遍历（DFS）树 `tree` 并通过函数 `f` 将遍历结果合并起来。

`tree` 是一棵存有 a 类型信息的树，树按以下规则递归定义 `(list root son_1 ... son_k)`，其中 `root` 为树根节点，`son_i` 为 `root` 节点的第 i 个儿子所在的子树。

比如 `'(8 (91 (77 (64))) (38) (65 (13 (66)) (61)) (23))` 代表以下树：



`f` 的类型为 $(a, b \text{ list}) \to b$，`(f x (list y_1 y_2 y_3))` 的意思是：若 `x` 的子树分别被 `fold-tree` 成了 `y_1 y_2 y_3`，那可通过 `f` 函数将其与 `x` 合并，得到一个新的类型为 b 的值。

提示：可以使用 `map` 等高阶函数实现。

A.2 sum-tree

利用 `fold-tree` 实现 `sum-tree`，`(sum-tree ntree)` 表示求 `Number` 类型树 `ntree` 上所有结点的权值和。

例如对于 A.1 中举例的树，其和为 506。

A.3 dfs

利用 `fold-tree` 实现 `dfs`，`(dfs tree)` 表示求树 `tree` 的 DFS 序，以列表形式返回。

注意我们总是先遍历子树 `son_1` ，最后遍历子树 `son_k`。例如对于 A.1 中举例的树，其 DFS 序为 `'(8 91 77 64 38 65 13 66 61 23)`。不难发现 DFS 序其实就是树的 `list` 表示去掉所有中间的括号后的样子。

提示：`(append l1 l2)` 可以连接两个列表，你不用过于担心时间复杂度。

A.4 height

利用 `fold-tree` 实现 `height`，`(height tree)` 表示求树 `tree` 的树高，单结点树的高为 1。

例如对于 A.1 中举例的树，其树高为 4。