Homework 5 - Meta-Circular Evaluator

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。

A m-eval

在 `m-eval.rkt` 中我们实现了一个最基本的元循环求值器，能解释的语言为 Racket 语言的一个子集，目前支持以下功能：

1. 基本类型，包括符号、字符串、数字；
2. `define`、`begin` 表达式的解释；
3. 最基本的 primitive procedures 及其 application，见代码的 `primitive-procedures`，你也可以自己添加。

它可以像 Racket 一样交互式解释：

```
racket m-eval-driver.rkt
```

也可以读入一个 Racket 程序解释执行：

```
racket m-eval-driver.rkt ./testcases/testcase-example.rkt
```

不过目前的解释器和真实的 Racket 解释器之间有一些微妙的差异，包括且不限于：

1. 定义顺序，见 SICP CH4.1.6；
2. 运行效率低，但我们的测试用例都很简单，你可以忽略这一点；
3. `define` 语句忽略了如 `(define (id x) x)` 这样的语法糖，我们的测试用例不会包含这种情况。

本次的作业中你需要：

1. 阅读并理解 `m-eval.rkt`，可以参考课堂 Slides 与书本 (SICP CH4.1.1 CH4.1.5)；
2. 添加 `if` 表达式的解释，无需添加对 `and` 和 `or` 的处理，用例中不包含这种情况；
3. 添加 `cond` 表达式的解释，可以参考书本 (SICP CH4.1.2)，书中给出了一种 `cond` 的实现方式；
4. 添加 `lambda` 表达式的解释，无需实现如 `(define (id x) x)` 这样的语法糖。

助教能力有限，如遇到 BUG 请及时反馈。

Reference

1. D. Friedman, M. Felleisen, *The little Schemer (4th ed.)*. MIT Press, 1996. [pdf]
2. H. Abelson, G. Sussman, *Structure and Interpretation of Computer Programs*. MIT Press, 1996. [pdf]