Lab - Crafting Interpreters

本次 Lab 的 Dead Line 暂定于 6 月 20 日（考试周最后一周的周五）晚 23 点 59 分 59 秒。
有问题可以通过 23110240130@m.fudan.edu.cn 邮件或在 elearning 上提问。

1 选题

本课程的大作业为给一个领域特定语言写一个解释器。目标语言自拟，以下给出一些例子参考：

1. 简单类型 λ 演算（SLTC）：用于执行和验证带有静态类型系统的 λ 演算表达式，其核心功能包括类型检查（确保函数应用和变量符合类型约束）和表达式求值；
2. 面向对象语言：管理类与对象的创建、继承关系、方法动态分派及封装等特性；
3. 硬件描述语言（HDL）：模拟数字电路行为，用于描述、模拟和验证硬件结构；
4. 逻辑型程序设计语言（类似 Prolog 语言）：基于形式逻辑进行推理，通过事实和规则构建知识库，支持用户查询，通过模式匹配推导答案，适用于符号计算与逻辑问题求解。

如果上述选题不满意你可以选择自拟。

2 交付物

你需要交付解释器的实现代码和相关文档，其中实现代码要求有：

1. 目标语言解释器，必须使用 Racket 实现；
2. 若干目标语言代码，作为测试用例；

文档需要包含：

1. 目标语言的功能性说明，包括你预计实现的功能点和设计亮点（需体现工作量）；
2. 目标语言的语法描述，形式不限，推荐基于 S-Expression；
3. 语言和解释器的设计与实现细节，关键功能的实现讲清楚即可，不需要太长篇幅；
4. 测试用例说明，用于验证你的功能点的确完成了；
5. 运行指南，需要确保助教可以在自己的机器上运行起来（macOS、Windows 11、Ubuntu 22.04 三选一），如有额外需要安装的库请告知。

3 评价指标

1. 基础功能实现 70 分：解释器能正确运行、核心功能完整且正确、有一定工作量；
2. 测试用例 10 分：能覆盖目标语言的核心功能与一些边界情况；
3. 代码质量 10 分：解释器架构设计、代码规范、错误处理等；
4. 文档 10 分：目标语言功能和语法说明完整（助教可以根据文档用你设计的语言写出程序）、运行指南清晰；

5. 额外加分项：语言设计新颖、有高级功能（比如 SLTC 中实现了部分的类型推导或类型检查等）。

最终得分为 min(100, 实际得分)。

4 有什么例子可以让我参考一下？

以下给出一些例子，仅供参考：

1. HW5 - Meta-Circular Evaluator；
2. HW6 - Object-Oriented Evaluator，工作量请略高于它；
3. 本文档的附件 `fdlang-eval.rkt` 实现了一个操作字符串的玩具语言解释器 `fdlang`，并有该语言的一个程序 `sample.fdlang`。你可以通过 `racket fdlang-eval.rkt sample.fdlang` 运行它。你也可以参考它作为你的起始代码。

注意，如果你的目标语言和解释器实现都与上述某个例子过于相似（也就是说你基本上是复制黏贴改了一小改），你将不得分。