本 lab 是在windows 11 环境下，Racket 版本v8.15的编译环境下完成的

example1 为语言正常运行时的案例

example2 为类型检查不合法时该语言给出报错信息的案例

运行测试样例的指令为在命令行输入 `Racket compiler.rkt <filename>` 。

下面分别给出我本机在example1和example2的运行结果以供参考：

```
>racket compiler.rkt example1.lang
=== 测试1: 基础类型兼容性 ===
动态+静态运算结果: 52
处理数字: 101
处理字符串: hello
=== 测试2: 函数类型演进 ===
动态版本: 30
部分类型版本: 30
静态版本: 15
=== 测试3: 复合数据类型处理 ===
动态列表处理: (2 hello! 4 world! 6)
类型约束版本: (2 hello! 4 world! 6)
=== 测试4: 类型转换和检查 ===
安全除法: 10/3
类型感知处理(整数): 25
类型感知处理(字符串): test processed
=== 测试5: 渐进式重构演示 ===
动态版本结果: ((Alice 85 B) (Bob 91 A) (Charlie 247/3 B))
部分类型化版本结果: ((Alice 85 B) (Bob 91 A) (Charlie 247/3 B))
=== 测试6: 类型安全演示 ===
安全字符串操作: Count: 42
灵活加法(正确): 30
高阶函数应用: 7
=== 测试7: 类型边界测试 ===
类型推导结果: (Int String (List ?) (? -> ?))
联合类型处理(Int): 142
联合类型处理(String): 4
联合类型处理(Bool): 1
=== 测试8: 配置文件处理器 ===
动态配置处理: Server: localhost:8080 (SSL)
类型安全配置处理: Server: localhost:8080 (SSL)
=== 测试9: 简单类型推断 ===
const1: (Int Int -> String)
add1: (Int Int -> Int)
add2: (Int Int -> Int)
comp1: (Int Int -> Int)
div1: (Int Int -> Number)
mix1: (Int Int -> Number)
=== 测试完成 ===
```

```
>racket compiler.rkt example2.lang
=== 测试1: 变量定义类型不匹配错误 ===
正确定义 x: 42
=== 测试2: 函数类型的递归检查 ===
函数类型推断输出:
```

```
((Int -> Int) Int -> ?)
(Int -> ?)
(String -> ?)
(Int -> String)
正确输出：1
=== 测试3: Cast操作失败错误 ===
正确的Cast操作: 42
=== 测试4: 联合类型边界错误 ===
联合类型处理(Int): number
联合类型处理(String): string
=== 测试5: Let绑定中的类型错误 ===
正确的Let绑定: x : 10, y : hello
=== 测试6: 高阶函数类型匹配错误 ===
正确的map使用: (2 4 6)
=== 类型错误测试完成（注：错误用例已注释以防程序中断）===
取消注释相应行可以观察具体的类型错误信息
```