COMP 520 Compiler Design
Group Milestone #2

Symbol Table and Type Checking for GoLite

Due: Sunday, March 17, 11:59 PM

Overview

The purpose of this milestone is to finish the front-end of your project. After this milestone you should have all the infrastructure ready to generate code.

Question 1: *Example Programs* (10 points)

Write the following example programs ending with file extension `.go`

1. **Type-Incorrect Example Programs** (10 points)

   Write 10 incorrect `GoLite` programs per team member which each exhibit a *different* type-checking or semantic error. Ideally, each program should be minimally sufficient to trigger the error, so think carefully about your program. Include a comment at the start of each file describing the intended issue.

Note that although we only require a small set of example programs for this question, you should prepare a more substantial test bank for debugging your project. As part of evaluating your work, we will execute our own comprehensive test suite that covers all language features.

Question 2: *Symbol Table and Typechecker* (30 points)

Implement the symbol table and type checker for `GoLite`. The scopes and type system for GoLite is the same as for Go, except on our restricted language subset. The typing and semantics rules were defined by Vincent Foley and are available at `www.cs.mcgill.ca/~cs520/2019/project/Milestone2_Specification.pdf`

For this milestone you must implement 2 compiler modes:

- `symbol`: Outputs the symbol table to `stdout` showing the declarations that are added in each scope as well as their kind/type (see the reference compiler for more details). If an error occurs, write the message to `stderr` and exit with status code 1 (note that the symbol table might be incomplete)
- `typecheck`: Outputs `OK` if the input is type correct, or an appropriate error message

Your front-end should handle semantic/type errors in a user-friendly way. You only need to catch the first error and exit, but you should try to give a reasonable error message that would help the user correct their program.

Question 3: *Design Decisions and Contributions* (10 points)

Briefly discuss the design decisions you took in the design and implementation of your symbol table/type checker. If there are parsing issues that you implemented as a weeding phase, document them here. Also include in this discussion

- An overview of the scoping rules you used;
- For each invalid program, the corresponding typing rule;
- Summarize how your team is organized and what each team member contributed.

Note: You should also keep notes on each phase, as this will help you generate the final project report.

What to hand in

Create a tag in your Github repository named *milestone2* (lowercase, no extra characters). Information about creating git tags can be found at: `http://git-scm.com/book/en/v2/Git-Basics-Tagging`. Your project should be kept in the following format

```
/
    README   (Names, student IDs, any special directions for the TAs)
    programs/
        2-typecheck
            invalid/ (incorrect programs)
    doc/     (Design documents)
        milestone2.pdf
    src/     (Source code and build files)
    build.sh (Updated build script)
    run.sh   (Updated run script)
```