COMP 520 Compiler Design
Group Milestone #2

Symbol Table and Type Checking for GoLite

Due: Monday, March 13 at 12:00 PM (lunch time)

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: *Invalid Programs* (10 points)

Develop 20 small example programs that illustrate each of your type/semantic errors. In Question 3 you will explain the type check that goes with each of these programs. We will make a library of these programs for further testing.

You should test your compiler on many more programs, both valid and invalid, but we will not grade these. Having a well tested compiler at this point will greatly simplify generating good code in the next milestones.

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.

Your compiler should support two new flags:

-dumpsymtab This flag should cause the top-most frame of the symbol table to be dumped each time a scope is exited. It would make sense to also print the line number associated with the scope exit. You may optionally implement `-dumpsymtaball` which dumps the entire stack of frames on each scope exit. Given an input file of the form `foo.go`, your compiler should write these results to `foo.symtab`.

-pptype This flag should pretty print the program, with the type of each expression printed in some legible format. It was suggested that you might print the types in comments so that the resulting pretty-printed output could be recompiled. Given an input file of the form `foo.go`, your compiler should write the results to `foo.pptype.go`.

Your front-end should handle semantic/type errors in a user-friendly way. You need to only catch the first error and then quit, but you should try to give a reasonable error message that would help the user correct his/her program. Error messages should be sent to stderr. As in the minilang assignments, we will be using test scripts to evaluate your submissions.

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

Briefly discuss the design decisions you took in the design and implementation of your symbol table and type checker. Briefly explain the scoping rules you used. Also, enumerate the type checks that you made, and give the correspondence of this type check with the test program used to demonstrate the type error given in Question 1.

Briefly list the major contributions made by each team member to this milestone.

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

What to hand in

You will be developing your project in your team's github repository. At each milestone you will create a tag before the due date, and the TAs will review the code associated with that tag. For this second milestone you should create a tag called *milestone2*. 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          (Your group names, student IDs, relevant info and
                     instructions for each milestone (just add information
                     as you finish each milestone. Make it easy for the TAs to
                     grade your milestone! Your README file should also
                     include references to any code that you have read to get
                     ideas,  or code that you have used in your compiler.
                     For code that you have used, you must ensure that:
                        (1) you have permission to use it, and
                        (2) you have clearly indicated in your compiler
                            the source of the code.)
    programs/
        valid/      (your valid programs)
        invalid/    (your invalid programs)
            types/  (your invalid types programs corresponding to Q1)
    src/            (the source code and build files.  You must use some sort of
                     automatic build system like Makefile or ant)
    doc/            (design documents, the answer for question 3 should be in a
                     file called milestone2.pdf)
    build.sh        (a build script to compiler your compiler)
```

```
run.sh          (a run script that when invoked as "run derp.go" will run
                 your compiler)
```