COMP 520 Compiler Design
Group Milestone #1

Scanner, Parser and Pretty Printer for GoLite

Due: Friday, February 21, 11:59 PM

The first milestone of the `GoLite` project covers the lexical, syntactic, and intermediate representation of your compiler. Just like your assignments, you may use any toolchain you wish (in class we showed both flex/bison in C and SableCC in Java) provided that it runs on the SOCS Trottier machines. You may also hand-code a scanner and/or parser if you feel ambitious!

The language specification was defined by Vincent Foley and is available at `www.cs.mcgill.ca/~cs520/2020/project/Milestone1_Specifications.pdf`.

Question 1: *Example Programs* (10 points)

Design a test suite for your compiler, writing the following programs with extension `.go`

1. **Syntactically-Correct Example Programs** (5 points)

   2 syntactically correct `GoLite` programs per team member that perform an *interesting* computation. Use a variety of language features!

2. **Syntactically-Incorrect Example Programs** (5 points)

   10 incorrect `GoLite` programs per team member that exhibit a *different* scanning or parsing error. Each program should be minimally sufficient to trigger the error, so think carefully. Include a comment at the start of each file describing the intended issue.

Note: 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: *Scanner, Parser and Pretty Printer* (30 points)

Implement the scanner, parser, AST, and pretty printer for `GoLite` using your chosen toolchain. As noted in the document, some weeding passes are also required as part of this milestone.

For the first milestone, your compiler must support 4 modes supplied as a command-line argument:

- `scan`: Outputs `OK` if the input is lexically correct, or an appropriate error message
- `tokens`: Outputs the token kinds, one per line, until the end of file. Tokens with associated data (literals, identifiers, etc) should be printed with their respective information
- `parse`: Outputs `OK` if the input is syntactically correct, or an appropriate error message
- `pretty`: Outputs the pretty printed code

Normal output (`OK`, tokens, pretty) must be sent to `stdout` and your compiler must exit with status code 0. On error, the message must be sent to `stderr` and your compiler must exit immediately with status code 1. Error messages should be descriptive (look into error reporting in your toolchain) and formatted "`Error: <description>`". Follow the output specifications **exactly**.

Scripts

Your project must follow the directory structure provided by the GitHub repository: `https://github.com/comp520/Assignment-Template`. In your submission, provide the following 2 scripts:

- `build.sh`: Builds your compiler using `Make` or similar
- `run.sh`: Takes two arguments (mode and input file) and executes your compiler binary (i.e. `./run.sh <mode> <filename>`)

Comments found in both files provide the exact requirements. A convenience `test.sh` script executes all programs in the `programs` subdirectories and reports any issues found.

Question 3: *Design Decisions and Team Work* (10 points)

Write a (maximum 5) page report, discussing the design decisions you took in the design and implementation of your scanner, parser, AST, and pretty-printer. If there are parsing issues that you implemented as a weeding phase, document them here. Also include in this discussion:

- The rationale of the implementation tools and language that you chose;
- Summarize how your team is organized and what each team member contributed; and
- Resources consulted (other students, websites, repositories, etc.). If no resources were consulted, please state "We worked alone". All code/programs must represent your own efforts – please check with us if in doubt, and consult the GoLite project slides.

Submission

Create a tag in your Github repository named *milestone1* (`http://git-scm.com/book/en/v2/Git-Basics-Tagging`). Your project must be kept in the following format:

```
/
    README   (Names, student IDs, any special directions for the TAs)
    programs/
        1-scan+parse/
            valid/   (correct programs)
            invalid/ (incorrect programs)
    doc/     (Design documents)
        milestone1.pdf
    src/     (Source code and build files)
    build.sh (Updated build script)
    run.sh   (Updated run script)
```