# Comp 520: Milestone 1

The Heapsters

February 2016

## Team Organization and Contributions

### General

Overall, some team members focused more on implementation and adding to the GitHub repository while others focused on debating the details of our context-free grammar and abstract syntax tree. We had seven group meetings over the course of this milestone which lasted an hour or more (often more than three or four hours). All group members were present during these meetings.

Additionally, some tasks were handled by all members:

- Everyone wrote tests (the number of tests varied by group member)
- Each member had a say in the design of the grammar and abstract syntax tree. Conflicting opinions were frequently worked out until all three members were in agreement.
- In many cases, changes to the grammar, abstract syntax tree, or sablecc document were made as a result of group discussions
- Everyone reviewed this document before it was submitted
- Hardik came up with the name. Leo and Ethan approved it.
- Everyone participated in creating the group handshake. Hardik designed it, Leo implemented it, and Ethan critiqued it by shaking his head in disappointment.

### Leo

- Designed the context-free grammar with Ethan on a whiteboard
- Wrote the pretty printer
- Modified the abstract syntax tree/context-free grammar while writing the pretty printer

### Hardik

- Designed the tokens that we wound up using for our sablecc document
- Implemented the lexer for semicolon insertion
- Implemented the context-free grammar in our sablecc document and made changes as necessary
- Implemented the abstract syntax tree in our sablecc document
- Consolidated our tests from individual branches of our GitHub repository and put them into the master branch

### Ethan

- Designed the context-free grammar with Leo on a whiteboard
- Wrote this document
- Kept minutes for our meetings

## Design Decisions

Our group decided to build a GoLite compiler in Java using SableCC 3. We decided on building a GoLite compiler because the specifications were most clearly defined, and built it in Java so we could devote more time to designing the grammar and abstract syntax tree instead of spending this time on the implementation details of writing a compiler in C. We decided to use SableCC 3 since both Ethan and Leo had used it for their previous assignments and found it to be a useful tool for parsing and syntax tree building.

We worked on this milestone iteratively. First, we each wrote at least ten tests for the scanner. Hardik created a token list that we modified as necessary. For the next few meetings we focused on creating a working context-free grammar. Ethan & Leo came up with one on the board, while Hardik added sections of the grammar to the SableCC file until the entire grammar compiled. After this, the three of us worked together on the abstract syntax tree. Next, we focused on writing more tests while Leo wrote the pretty printer and Ethan wrote the report. On the last few days we met and resolved any outstanding issues.

In order to focus on writing more tests, we decided not to implement any weeding phases until the next milestone. We did, however, include certain constraints within our grammar to reduce the amount of weeding required (e.g., we limited the set of allowed top-level statements). Some issues that we chose not to handle include the removal of quotes/backticks from strings, checking proper usage of breaks and continues, checking that fields and array accesses happen on the correct elements, checking for string casts, checking that expressions in simple stmt are valid (e.g., function calls), mismatching numbers of ids and

values in declaration statements, and distinguishing between function calls and custom type casts. In the weeding phase we may also consider constraining the package declaration to main and checking to make sure one main function exists in each file.