CS 153 / SE 153 / CS 253

Concepts of Compiler Design

Fall Semester 2012

Department of Computer Science
San Jose State University
Instructor: Ron Mak

Assignment #6

Assigned: Wednesday, October 24
Due: Friday, November 2 at 11:59 pm
Team assignment, 100 points max

Generate a parser with JavaCC

Design a grammar for your source language and write its productions rules using EBNF. Use the EBNF as a guide to create a **`.jj`** file that includes the production rules and any required token specifications. Use JavaCC to generate a working parser and scanner for your source language.

This assignment is the beginning of your team compiler project. Your compiler projects are due Friday, December 14.

What to turn in

- A text file containing at least 20 EBNF production rules for your source language. (Tip: Use JJDoc.)
- The **`.jj`** file.
- A sample program (or statements) written in your source language that your parser can successfully parse. This program should show off the capabilities of your parser as far as you implemented it for this assignment. For this assignment, your parser does not have to do error handling (the input can be error-free).

Your parser should print out informative messages as it recognizes the major source language constructs (such as statements and expressions) in the source program. Your parser does not need to produce fancy output.

Email the files (you can zip them) to ron.mak@sjsu.edu. In the subject line, include ASSIGNMENT #6 and the name of your team.

Your source language

You may choose a source language for your team project to compile and execute:

- It must be a procedural language.
- It can be an existing language or subset thereof.
- If you choose Pascal, then your completed project *must include features not in the WCI book*, such as statement labels and **`GOTO`** statements, **`WITH`** statements, set types and expressions, pointer types and expressions, etc.
- You can invent a new language, as long as it's procedural (no Lisp or Lisp-like languages). **CS 253 students must invent a new language.**
- By the end of the semester, you must implement enough of your source language to be able to execute nontrivial programs written in it.

Keep your source language and its grammar **simple for this assignment**! This is a snapshot of your early thinking about language design. You can change or add more features later. Suggested order of implementation for your grammar and compiler:

- Expressions with numeric constants and scalar variables (no type checking, no arrays or records yet)
- Assignment statements
- Control statements
- Variable declarations (no type definitions yet)
- Procedure and function declarations
- Procedure and function calls
- Type definitions

Do at least the first three for this assignment. You don't have to do all your control statements; you can add more later.

As you incrementally implement more of your language, write small test programs to ensure that each new feature works. To verify that you didn't break anything, do regression testing by re-running the test programs you had written earlier.