San José State University
Department of Computer Science

CS 153
Concepts of Compiler Design

Fall 2017
Instructor: Ron Mak

Assignment #5

Assigned: Tuesday, October 24
Due: Friday, November 3 at 11:59 pm
Team assignment, 100 points max

ANTLR 4 grammar

Write the first draft of the grammar for your compiler project. Create a **`.g4`** grammar file and use ANTLR 4 to generate a working parser and lexer (scanner) for your source language. Generate syntax diagrams from your grammar and parse trees from sample programs written in your language.

This assignment is the beginning of your team compiler project. Your compiler projects are due Monday, December 11.

Your source language

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

- 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 **`WITH`** statements, set types and expressions, pointer types and expressions, etc.
- You can invent a new language (highly recommended), as long as it's procedural. No Lisp or Lisp-like languages.
- 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:

- Regular expressions that define your language tokens.
- 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 five for this assignment. You don't have to do all your control statements; you can add more later.

Write at least three sample programs in your language.

Syntax diagrams

Generate syntax diagrams from your grammar file. You can generate the diagrams using the Eclipse plug-in. There are also standalone tools that can generate the diagrams. Google "ANTLR 4 syntax diagrams".

Parse trees

As you incrementally implement more of your language, write small test programs in your source language 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.

Use **`grun`** to generate and display GUI parse trees from at least three of your sample source programs. Choose the sample programs that show off the most features of your language as you've defined it so far. To use **`grun`**, you must first generate and compile the Java version of the parser and lexer. Successfully generate parse trees mean that your parser and lexer are working properly.

What to turn in

Submit to Canvas, **Assignment #5 ANTLR Grammar**, a zip file containing:

- A **`.g4`** grammar file.
- The generated syntax diagrams. These can be saved PDF or PNG files, or screen shots.
- GUI parse trees generated by **`grun`**, one each for at least three sample source programs. These can be saved PDF or PNG files, or screen shots.

Name the zip file after your team; for example: **`SuperCoders.zip`**

Rubric

Your program will be graded according to these criteria:

Criteria	Max points
• ANTLR 4 **`.g4`** grammar file.	• 40
• Generated syntax diagrams.	• 30
• Generated GUI parse trees for three sample source programs.	• 30