

Project 3

A syntax recognizer and semantic checker for C-- programs

DUE DATE: May 18, 2020

Part 1. Complete Production rules (50 pts)

In the template file `parser.y`, we have provided an incomplete set of C-- production rules. Your job is to fill the missing productions (they are marked by C comments, e.g. `/**/`) so that the parser you created through Yacc/Bison can parse correct C— programs.

Part 2. Build Abstract Syntax Tree (50 pts)

In a multi-pass compiler, we usually generate AST as a by-product of syntax checking. Several follow-up passes such as semantic checking, code generation and code optimization can be performed thereafter based on the AST. You are required to build such AST using the node structure we suggested/provided in the `header.h`. In the `parser.y` template file, you need to add action routines to build the AST. There are some sample AST building action routines already in place just for your reference. However, the majority of such actions are currently missing from the template `parser.y` file, marked by C comments, such as `/**/`, and you need to fill in the missing actions. The generated AST will be printed out to an `AST.gv` file via the `printGV()` call (this has been coded in the `parser.y` main function). TAs will check the correctness of the AST generated from your parser. `AST.gv` can be converted to png file with `graphviz` tool, (you can download from <http://www.graphviz.org/>), see `README-gv2png.txt`. That graphic layout could help you see the AST structures and debug your action routines more effectively.

Reference:

- The Lex & Yacc Page: Online Manual

<http://dinosaur.compilertools.net/yacc/index.html>

Note:

- a) In the `hw3` directory you may find the following files:
- | | |
|---------------------------------|--|
| 1) <code>src/lexer.l</code> | the sample lex program that you may start with |
| 2) <code>src/header.h</code> | contains AST data structures |
| 3) <code>src/Makefile</code> | |
| 4) <code>src/parser.y</code> | template YACC file with incomplete production rules |
| 5) <code>src/functions.c</code> | functions that can be used to generate the <code>AST.gv</code> |
| 5) <code>test/</code> | a test directory containing some sample tests |

Submission requirements:

- 1) DO NOT change the executable name (`parser`).
- 2) Compress all your files as **studentID_hw3.zip**. Then upload to Ceiba.
- 3) You are free to modify `Makefile`. But make sure your **make** command works correctly.

