Compiler Design Homework 4
Due Before Week 6 Class

Compiler Project (Due Week 6)

- Use **ANTLR4** and the **Cminus.g4 file** you produced last week to produce a.**C++ version** of Cminus. Turn **on** the listener flag and the visitor flag to get all the code.
- Add a C++ **symbol table** to the header, and **semantic predicates** for **semantic analysis**. You will only need to put into the symbol table those things that you will need for the semantic analysis and the IR generation.
- Write a C++ file with a **main routine**.
- **Print** the contents of the symbol table for the C-Input-0.txt.
- Produce error messages for the semantics described in Appendix A, and validate that the AST produced by the ANTLR4 parser is consistent with the Cminus language. It is OK to modify the grammar to satisfy semantic rules.
- **Add the following rule to the Cminus semantics**:

  The **'='** operator **shall assign** the results of the expression to the **var L-value** but **shall not** return a result for further use by an expression. In other words an assignment cannot appear in an 'if' test or a 'while' test, and you **cannot** have an equation like **{ A = (B = (C = 0) + 1) – 1; }.** (Even though it should currently parse!)
- Remember that the Cminus language has two functions provided by the environment, the **int input(void)** and the **void output(int)** functions. If you want you may also add a **void outputCR(void)** function and/or a **void output(string)** function. Whatever you make part of the language, be sure the semantics has it available in the symbol table. Also remember that you must provide the code for these functions.
- All the Cminus input files provided should run. Produce output files to turn in.
- Make up some test case input files to validate your semantics, put your name or your initials in the filename and **share your test files with everyone in the class** by placing them in the Blackboard: Discussions: HW4… Forum **by Week 5**. Your semantic analysis (or modified grammar) should catch all the problems in all the test code and produce error messages. Pipe your error messages to a file to turn in.
- Make a **zip** file and include a **readme** file and a directory called **CompilerProject** with your **Cminus code**, the **C++ code** it generates, the C++ code you wrote, your **input** and **output** files and the **symbol table** dump. **Submit results to Blackboard before the Week 6 Class.**

Spreadsheet Project (Due Week 6)

- Include **panic mode** error recovery into your parser routines.
- Provide **error messages** for the spreadsheet as **three lines** messages:
  1. Display the input line containing the error, (Note: be sure to use a proportional font like `Courier New`)
  2. Display a line **with ^ to point at the position of error** in the first line, (**Note**: be sure to use the same proportional font as line 1.) and
  3. Display the error message in its own line.
- Make some original input files with errors in them to test that the possible error conditions are reported correctly. Include your name or initials in the names of the test files. **Share your test files with everyone in the class** by placing them in the Blackboard: Discussions: HW4… Forum **by Week 5**.
- Submit a **zip** file containing a **readme** file, all the **code** you have written for the spreadsheet, the **executable** (identify the OS in the readme), the **input** and the **output** files you used to test your code.
- **Submit the zip file to Blackboard within two weeks before the Week 6 Class.**

Read Louden Chapter 6.0, 6.1.0 and 6.2.2–6.3.0 (Semantics)

Read ***The Definitive ANTLR4 Reference*** (DA4R) **10.0** [Attributes & Actions], DA4R **11.0** and **11.3** [Semantic Predicates]. Use DA4R 15 [the Reference Section] as needed.

Read the ***Language Implementation Patterns*** (LIP) 6.0 [Program Symbols] and 6.4 & 6.5 [Symbol Tables].

Read Kaleidoscope Tutorial (**not** the OCaml tutorial)
Chapter #1: Kaleidoscope language and Lexer,
Chapter #2: Implementing a Parser and AST and
Chapter #3: Code generation to LLVM IR.

Submit all results to Blackboard before the appropriate Class.