Lab 2

COEN 175 Compilers

Overview For Lab 2

Goal

- Write recursive-descent parser for Simple C that will print out the operation order of a given input program

Submission

- Submit a tarball of your cpps and make files
- Due Date: Sunday January 23rd

Main Objectives

* You will be given a working solution for phase 1
- Disambiguate your grammar by “hand”
- Modify `lexer.cpp`
- Create `parser.cpp`

Disambiguating Grammar

- Work on it for the first ~30 min
- Eliminate left recursion
    - Never try to match with itself before any operators
    - Achieve right recursion with “prime” expressions
- Left-factor
    - But don’t overdo it
- One function for each level of precedence

lexer.cpp

- Have the output of `lexan()` be an integer to indicate what the token is
- tokens.h provided for token output
- No logic change necessary, just change return values
- Utilize given working solution

parser.cpp

- A `report()` function will be given to you in `lexer.cpp`
  - Can help with debugging in parser
- Translate your disambiguated grammar into functions
- One function for each level of precedence
- Focus on expressions, terms, and factors this week

Tips

- Trust your disambiguation when you start writing code
- Don’t condense too much, it will work out better to be thorough than optimized
- Parenthesis can be a bit tricky, use an extra lookahead to help determine if it is the type cast or parenthesized expression case
- Your functions can be written recursively, but it might be easier to write them iteratively except for prefix expressions
- Postfix expressions can go forever but be careful what you keep parsing
- **READ THE SYNTAX RULES CAREFULLY**

Testing Code

- Include parser in given makefile
- Run `make`
- ./scc
- Input a complex expression with +, *, ()