Project #2

CS 3510 – Spring 2015

Nathan Harris and Jake Prem

I. Requirements: Create a Parser for the C- language

II. Design: Our first step in the design process was to fix the grammar from the book on page 492. This required some left recursion removal and left factoring. After this was done, we started to create our classes which would hold the data we needed to create an AST. We then started work on the bulk of the project which was the parseX() routines and the printTree routines.

III. Implementation: Although it is fairly straightforward coding once you understand what you are doing, I found creating the classes one of the most difficult parts of this project, especially since some of them did not correspond with the grammar rules. Another difficult part of this project was getting the first and follow sets exactly right in the code so that the parser would not pick up any false errors. In addition, the tremendous amount of recursion in this project created a tremendous amount of difficulty during debugging, especially in parseBinaryExpression(). I was also getting an infinite loop when I tried to print out a BinaryExpression, since I was assigning the lhs of a BinaryExpression in parseBinaryExpression to itself. Therefore, it would recursively assign lhs to itself infinitely.

IV. Testing: We tested our program using a simple quicksort algorithm and the example program from the text on pg 27. We also tested out program using the testcode.c file.

V. Summary/Conclusion: The program appears to function properly. It gives the correct answer for every input.

VI. Lessons Learned: I think it would have been a bit more helpful to finish our first and follow sets before diving into our code. We often found ourselves staring at our grammar for a few minutes at a time just to see what case we were in and where we should go next.

.

Our code compiled and ran properly, and produced the correct output