Semester Project – Part 2 – CSE244 – Fall 2000

Due: Thursday, November 2, 2000, at 2:00pm

The second part of the semester project focuses on gaining experience with lex (flex) and yacc (bison), coupled with continued design (and redesign) skills for context free grammars (CFGs). The second part of the project is divided into two major tasks, and is worth a total of 50 points:

1. Revising the common grammar to remove the shift/reduce and rule-not-reduced conflicts. These corrections are made to the yacc/bison file, and the result must be tested extensively for different input latex documents. This task is worth 35 points.

2. Using your result grammar of the first task, redesign the common grammar to support the recognition of verbatim blocks and also nested blocks. That is, recall that the different blocks (itemize, enumerate, single, etc.) can also be nested to yield different combinations of formatted text. Again, test your result using yacc/bison to demonstrate that the grammar revisions for these two changes work correctly. This task is worth 15 points.

This is again an individual project, to give each student the opportunity to work with and learn the compiling writing tools lex and yacc. While it is permissible to ask one another questions, all of the work should be your own.

To serve as a common basis for the project, the web page contains different files and directories for project, part 2. They are summarized below:

Files
latex.in : A sample input file.
latex.1 : A sample latex lex file.
latexp2.y : Contains a yacc specification with fprintfs for debugging.
Grammar most closely resembles the original case.
latexp2clean.y: Equivalent specification without the fprintfs.
projp2.tex : This file - serves as additional sample latex input.
projp2.ps : Postscript version of Project, part 2.
projp2.pdf : PDF version of Project, part 2.

Directories
Bison: Conflicts and Debugging information generated by Bison
Files: b2conflicts.txt: S/R and R/R Conflicts from Bison
bison.debug.txt: Short Overview of Bison -v Output
latexp2.output : Complete Bison -v Output
latexp2.tab.c : Parser Generated by Bison
lex.yy.c : Lexical Analyzer Generated by Flex
PCYacc: Conflicts and Debugging information generated by PCYacc
Files: yy.ltr.txt : Complete pcyacc -v Output - search for ? for S/R and R/R
    yytab.h  : Defines Generated by pcyacc -d
    latexp2.c : Parser Generated by pcyacc
    latex.c  : Lexical Analyzer Generated by plex

UnixYacc: Conflicts and Debugging information generated by Unix Yacc
Files: y2conflicts.txt: S/R and R/R Conflicts from Unix Yacc
    y.output   : Complete Unix Yacc -v Output
    y.tab.c    : Parser Generated by Unix Yacc
    lex.yy.c   : Lexical Analyzer Generated by Flex

Notes: All three directories used the same latexp2.y and latex.1 input files for yacc and lex. There are comments in each file explaining what that file contains.

Disclaimer: No guarantees are given regarding the "correctness" of each of these files. I did some basic testing to make sure that things, in general, would execute.

Note that intentional 'errors' have been placed in latex.in and latex.1. Note also that the grammar files contain numerous shift/reduce errors and other problems, as will be discussed in class.

The second part of the project is due on November 2, 2000, at 2:00pm. Please hand in the following:

1. The revised yacc/bison specification for task 1. DO NOT HAND IN "*.c" files!
2. A log file that documents the changes made to the grammar to eliminate the shift/reduce errors and other problems for task 1. Make sure that you provide both the original and revised grammar segments for each change that you make! Also include any remaining shift/reduce or reduce/reduce errors, but NOT the entire "output" file.
3. The revised yacc/bison specification for task 2. DO NOT HAND IN "*.c" files!
4. A log file that documents the changes made to your grammar of task 1 in support of verbatim and nested blocks. These changes may occur in both the lex(flex) and yacc(bison) files! Again, provide original/revised segments for each change, and remaining S/R or R/R errors.
5. Test cases and test results for both tasks, clearly marked and organized.
6. Detailed compilation instructions for your software. Please make it clear the platform (unix/pc) and language (C/C++) and version of lex/yacc that was utilized for your project.

Hand in your projects using an electronic media (via email - zip file) or on floppy disk. You will use this media for your demo. Please also provide hard copy of your yacc, lex, and log files.