CS126: Introduction to Compilers

Assignment 4: SYN1

Out: 9/26/03
Due: 10/3/03

1.0 Introduction

This assigment has two parts. The first part involves extending your JavaCC parser for Decaf so that it does some reasonable error handling. The second part is to experiment some with shift-reduce parsing by getting a grammar that bison (yacc) will accept without conflicts.

2.0 Error handling

You should go through the grammar and determine what are reasonable places to recover from errors. At these points you should handle ParseExceptions, add the appropriate message to the DecafErrorHandler, and attempt to recover and continue parsing.

You should also attempt to identify at least one common syntactic error and add specific grammar rules to detect it and print an appropriate error message.

3.0 LR Parsing

Your second task this week is to write a bison/yacc input file for DECAF. This file will include both the definitions of the various tokens you used in Assignment 2 as well as the definition of an appropriate grammar for DECAF that is acceptable to yacc or bison.

Your program doesn't have to run or compile. Instead, all you have to do is to get yacc or bison to accept your input file without any errors or warnings.

The grammar for DECAF is given in the handout "Syntax of the Brown Decaf Programming Language" that was distributed during the first week of class. You should be as true to that grammar as possible in creating your parser.

This assignment is not as easy as it looks. While you might start with a lot of simple typing to convert the grammar rules from the syntax in the handout into the syntax provided by yacc, you will soon find out that this alone is not sufficient. There are several places in which the given grammar is not LALR (or even LR(k)). Some of these (such as dangling else and expressions) are discussed briefly in the handout.

Others you will find are just there and produce various conflicts when you run yacc or bison.

You are going to have to take each of these conflicts, determine what causes it, and then modify your grammar so that it is eliminated. If at all possible (and it is possible), you should avoid requiring later semantic checks or processing for a successful parse.

To help you find the cause of conflicts, you will probably want to run yacc or bison with the -v option to produce a trace of what the parser generator thinks it is doing. You should be able to work backwards within this trace to find the sources of conflicts. Once you understand the source of a problem, you should be able to modify the grammar so that the problem goes away. This will involve merging and splitting rules as appropriate and probably a bit of experimentation. It will also force you to understand the process whereby LALR tables are generated.