Phase 2: Parser Generator

Parser Generator

Objective

This phase of the assignment aims to practice techniques for building automatic parser generator tools.

Description

Your task in this phase of the assignment is to design and implement an LL (1) parser generator tool.
The parser generator expects an LL (1) grammar as input. It should compute First and Follow sets and uses them to construct a predictive parsing table for the grammar.

The table is to be used to drive a predictive top-down parser. If the input grammar is not LL (1), an appropriate error message should be produced.
The generated parser is required to produce some representation of the leftmost derivation for a correct input. If an error is encountered, a panic-mode error recovery routine is to be called to print an error message and to resume parsing.
The parser generator is required to be tested using the given context free grammar of a small subset of Java. Of course, you have to modify the grammar to allow predictive parsing.
Combine the lexical analyzer generated in phase1 and parser such that the lexical analyzer is to be called by the parser to find the next token. Use the simple program given in phase 1 to test the combined lexical analyzer and parser.

Bonus Task

Automatically eliminating grammar left recursion and performing left factoring before generating the parser will be considered a bonus work.

Java CFG

```
METHOD_BODY:: =   STATEMENT_LIST
STATEMENT_LIST::= STATEMENT |   STATEMENT_LIST   STATEMENT
STATMENT ::=      DECLARATION
                | IF
                | WHILE
                | ASSIGNMENT
DECLARATION ::= PRIMITIVE_TYPE IDENTIFIER;
PRIMITIVE_TYPE ::= int | float
IF ::= if ( EXPRESSION ) { STATEMENT} else { STATEMENT }
WHILE ::= while ( EXPRESSION ) { STATEMENT }
ASSIGNMENT ::= IDENTIFIER = EXPRESSION;
EXPRESSION ::= NUMBER
                | EXPRESSION INFIX_OPERATOR EXPRESSION
                | IDENTIFIER
                | ( EXPRESSION )
INFIX_OPERATOR ::= + | - | * | / | %| < | > | <= | >= | == | !=| |   | &&
```

CFG Input File Format

- CFG input file is a text file.
- Production rules are lines in the form LHS ::= RHS
- Production rule can be expanded over many lines.
- Terminal symbols are enclosed in single quotes.
- \L represents Lambda symbol.
- The symbol | is used in RHS of production rules with the meaning discussed in class.
- Any reserved symbol needed to be used within the language, is preceded by an escape backslash character.

Input file example:

```
# METHOD_BODY = STATEMENT_LIST
# STATEMENT_LIST = STATEMENT |   STATEMENT_LIST   STATEMENT
# STATEMENT =     DECLARATION
| IF
| WHILE
| ASSIGNMENT
# DECLARATION = PRIMITIVE_TYPE ‘id’ ‘;’
# PRIMITIVE_TYPE = ‘int’ | ‘float’
# IF = ‘if’ ‘(‘ EXPRESSION ‘)’ ‘{‘ STATEMENT ’}’ ‘else’ ‘{’ STATEMENT ‘}’
# WHILE = ‘while’ ‘(‘ EXPRESSION ‘)’ ‘{‘ STATEMENT ‘}’
# ASSIGNMENT = ‘id’ ‘=’ EXPRESSION ‘;’
# EXPRESSION =    SIMPLE_EXPRESSION
| SIMPLE_EXPRESSION ‘relop’ SIMPLE_EXPRESSION
# SIMPLE_EXPRESSION = TERM | SIGN TERM | SIMPLE_EXPRESSION ‘addop’ TERM
# TERM = FACTOR | TERM ‘mulop’ FACTOR
```

FACTOR = ‘id’ | ‘num’ | ‘(‘ EXPRESSION ‘)’
SIGN = ‘+’ | ‘-‘

Parser Output File Format

Your program should output the predictive parsing table of the generated parser in a format of your choice as well as the leftmost derivation sententials one per line (like the following format).

Output file example for the given test program:

```
int x;
x = 5;
if (x > 2)
{
	x = 0;
}
```

METHOD_BODY
STATEMENT_LIST
STATEMENT_LIST STATEMENT
STATEMENT_LIST STATEMENT STATEMENT
STATEMENT STATEMENT STATEMENT
DECLARATION STATEMENT STATEMENT
PRIMITIVE_TYPE IDENTIFIER; STATEMENT STATEMENT
int IDENTIFIER; STATEMENT STATEMENT
….to be continued

Notes

1. Due date: Wednesday, April 27th 2016 at 11:59 PM.
2. Implement the project using C++.
3. Each group consists of 4 students.
4. Each group must submit the following to the online submission system "http://alexcs.noip.me/CSBox" under Dropbox -> CSBox -> Third -> Compilers -> deliverable02:

   1- Your executables and source code
   2- A project report: make sure that your report contains at least the following:
      a. A description of used data structures.
      b. All algorithms and techniques used
      c. Transition Diagrams if any
      d. Parsing Tables if any
      e. Comments about used tools
      f. Explanation of functions of all phases
      g. Any assumptions made and their justification.

Grading Policies

- Delivering a copy will be awful y penalized for both parties, so delivering nothing is so much better than delivering a copy.