Compilers Design: LL parser

Spring 2018

Programming Assignment 2

Due Date: June 19, 2018

You are going to write a LL(1) parser (you should write an LL(1) parser, not another parser in this assignment) for a toy programming language whose grammar is defined below. The grammar of this toy programming language:

program → compoundstmt
stmt → ifstmt | whilestmt | assgstmt | compoundstmt
compoundstmt → **{** stmts **}**
stmts → stmt stmts | ε
ifstmt → **if** **(** boolexpr **)** **then** stmt **else** stmt
whilestmt → **while** **(** boolexpr **)** stmt
assgstmt → **ID** = arithexpr **;**
boolexpr → arithexpr boolop arithexpr
boolop → **<** | **>** | **<=** | **>=** | **==**
arithexpr → multexpr arithexprprime
arithexprprime → **+** multexpr arithexprprime | **-** multexpr arithexprprime | ε
multexpr → simpleexpr multexprprime
multexprprime → ***** simpleexpr multexprprime | **/** simpleexpr multexprprime | ε
simpleexpr → **ID** | **NUM** | **(** arithexpr **)**

In this grammar, *program* is the start symbol. You may assume that each token is separated with at least one white space character (for simple reading). **ID** and **NUM** are also tokens (you do not have use actual identifier or numbers, just use these names). You should make sure that this grammar is suitable for the LL(1) parsing (I think it is. If it is not, you should solve the problem).

Before you create the parser, you should create the LL(1) parsing table of the grammar above. Your parser should read a program, and it should create and print the parse table of that program if it is a correct program. If the program is incorrect (contains syntax errors), it should print syntax error messages (together with line numbers). Your program should be able to recover after a syntax error, and continue to parsing.

Test your LL(1) parser with correct and incorrect programs. The output of a test with a correct program should be the parse table of that program; and the output of a test with an incorrect program should be a sequence of error messages (together with line numbers). For example, the output of your parser for the following correct program

```
{ ID = NUM ; }
```

can be the following parse tree:

```
program
  compoundstmt
    {
    stmts
      stmt
        assgstmt
          ID
          =
          arithexpr
            multexpr
              simpleexpr
                NUM
              multexprprime
                ε
            arithexprprime
              ε
          ;
      stmts
```

```
            ε
    }
```

You should hand-in:

- Your source program
- The LL(1) parsing table of the grammar
- Some test runs of your program with correct and incorrect programs
- Above all email to xmju@sei.ecnu.edu.cn.

DO IT YOURSELF – CHEATING WILL BE PUNISHED