Project 2
Implementation of a Recursive Decent Parser
Due Wednesday, May 1, 2019

1. Problem

In this assignment you are requested to use the tool ANTLR to implement a recursive decent parser for the simple language Rose. The grammar for Rose is given as follows:

program : **procedure identifier is declare** *variables* **begin** *statements* **end ;**
variables : *variables variable* | ε
variable : **identifier : integer ;**
statements : *statements statement* | ε
statement : *assignment_statement*
| *if_statement*
| *for_statement*
| *exit_statement*
| *read_statement*
| *write_statement*
assignment_statement : **identifier :=** *arith_expression* **;**
if_statement : **if** *bool_expression* **then** *statements* **end if ;**
| **if** *bool_expression* **then** *statements* **else** *statements* **end if ;**
for_statement :
for identifier in *arith_expression* **..** *arith_expression* **loop** *statements* **end loop ;**
exit_statement : **exit ;**
read_statement : **read identifier ;**
write_statement : **write** *arith_expression* **;**
bool_expression : *bool_expression* **||** *bool_term* | *bool_term*
bool_term : *bool_term* **&&** *bool_factor* | *bool_factor*
bool_factor : **!** *bool_primary* | *bool_primary*
bool_primary : *arith_expression relation_op arith_expression*
relation_op : **=** | **<>** | **>** | **>=** | **<** | **<=**
arith_expression : *arith_expression* **+** *arith_term*
| *arith_expression* **-** *arith_term*
| *arith_term*
arith_term : *arith_term* ***** *arith_factor*
| *arith_term* **/** *arith_factor*
| *arith_term* **%** *arith_factor*
| *arith_factor*
arith_factor : **-** *arith_primary* | *arith_primary*

arith_primary : **constant** | **identifier** | **(** *arith_expression* **)**

The above grammar is not LL(1). You need to transform it into an LL(1) grammar using left recursion elimination and left factoring.

You can follow the following steps:

1. Modify the grammar Rose.g4 in project 1 by replacing the parser rule 'token' by parser rules derived from the given grammar above as follows.

   // The grammar for Rose language
   grammar Rose;

   // Parser rules
   program : PROCEDURE ID IS DECLARE variables BEGIN statements END SEMI
   variables : …
   …
   // lexer rules
   BEGIN : 'begin'
   …

2. Use the ANTLR tool to generate the lexer and parser java code.

   $antlr4 Rose.g4

3. Compile the generated java code.

   $javac Rose*.java

4. Use the ANTLR tool to execute the lexer and parser.

   $grun Rose program -tree

A sample Rose program is given as follows:

```
// A sample Rose program
procedure EMPTY is
declare
begin
end ;
```

The output of the parser for this sample program is as follows:

(program procedure EMPTY is declare variables begin statements end ;)

2. Handing in your program

To turn in the assignment, upload a compressed file proj2 containing Rose.g, Rose.tokens, Rose*.java, and Rose*.class to Ecourse site.

3. Grading

The grading is based on the correctness of your program. The correctness will be tested by a set of test cases designed by the instructor and teaching assistants.