Homework 2 (Posted 1/25, Due 2/1, 11:59pm, on Gradescope)

IMPORTANT NOTE: All requirements stated for HW1 must be followed in this homework also. In all grammars given, **the nonterminal appearing in the first production rule** is the start nonterminal, unless specified otherwise. Grammar symbols that do not appear in the left-hand side of any production rule should be treated as a terminal for that problem. Use $ to indicate the end of input.

In this course, we use the brackets [] instead of the parentheses () when showing the FIRST set for a production rule, e.g. FIRST[1] for rule 1, etc. Use the parentheses () when showing the FIRST set and the FOLLOW set for a nonterminal, e.g. FIRST(A) and FOLLOW(A).

Problem 1 (10 pts) The following rules are written in extended BNF. Rewrite them in the basic (strict) BNF format, using as few rules as you can. The accepted input must be the same before and after the conversion. Please **number the individual rules** after the conversion.

[**Reminder:** Meta symbols in extended BNF are (,), *, and ?. These characters, when double quoted, are part of the terminals in the grammar.]

ClassDecl → class id { VarDecl* MethodDecl* }

VarDecl → Type id (= Exp)? (, id (= Exp)?)* ;

MethodDecl → public Type id “(“ FormalList? “)” { Statement* }

FormalList → Type id (, Type id)*

Problem 2 (30 pts) In the following grammar, **S** is the start nonterminal. Please check to see whether it is an LL(1) grammar by performing the following tasks. You are only required to list the final result. However, if you show the intermediate result of each iteration, it may help you obtain partial credit in case of errors.

(1) **List the FIRST set for each production rule’s righthand side and the FIRST set for each nonterminal**.

(2) **Compute the FOLLOW set for each nonterminal**.

(3) **Try to draw the LL(1) parsing table** and circle the parsing conflicts found in the table if any.

1. S → L = R ;

2. S → R ;

3. L → ID

4. L → *R

5. R → L

Problem 3 (25 pts)

Consider the following grammar:

1. VarDecl $\rightarrow$ Type VarList ;
2. VarList $\rightarrow$ VarItem
3. VarList $\rightarrow$ VarList , VarItem
4. VarItem $\rightarrow$ ID
5. VarItem $\rightarrow$ ID = num

Problem 3.1 (15 pts)

Make a list of FIRST sets and FOLLOW sets in the following order. You are only required to list the final result. However, if you show the intermediate result of each iteration, it may help you obtain partial credit in case of errors.

- List the FIRST set for each production rule's righthand side, denoted by FIRST[1] for rule 1, and so on.
- List the FIRST set for each nonterminal.
- List the FOLLOW set for each nonterminal.

Problem 3.2 (10 pts)

Try to draw the LL(1) parsing table based on result from Problem 3.1 and circle the parsing conflicts found in the table if any.

Problem 4 (25 pts)

Problem 4.1 (10 pts)

Convert the grammar in Problem 3 into an equivalent one that is LL(1), using as few production rules as you can.

Problem 4.2 (5 pts)

Repeat Problem 3.1, but use the new grammar written in Problem 4.1.

Problem 4.3 (5 Pts)

Repeat Problem 3.2, but use the new grammar written in Problem 4.1.

Problem 4.4 (5 Pts)

Follow the LL(1) parsing table constructed in Problem 4.3, Write a leftmost derivation for the input Type ID = num, ID ; (If no derivation exists to this input, write the leftmost derivation as far as you can, till you cannot proceed.)

Problem 5 (10 pts)

Consider the following grammar:

1. Term→ Term * Value
2. | Term / Value
3. | Term % Value
4. | Value
5. Value→ ! Factor
6. | Factor

Convert this grammar into an equivalent one that is LL(1), using as few production rules as you can. Next, draw the LL(1) parsing table. (You do not need to show intermediate steps that lead to the LL(1) parsing table.)