Homework 2: Solution
Spring 2013 Programming Language Constructs
Lecturer: Chi-Jen Wu (cjwu @ iis.sinica.edu.tw)
TA: Yi-Chan Kao (dogkevin @ hotmail.com)

Out: 3/21/13
Due: 3/28/13

This assignment is a handwritten homework.

1 Problems

Task 1. Please prove or disprove that an if statement could be replaced by a while statement in a programming language.

Solution: The principal difference between an if construct and a while construct is that if a condition is satisfied, the former executes a statement once, whereas the latter could execute it multiple times. In order to enforce that the while construct executes a statement precisely once, when a condition is satisfied, we use the following strategy: Let cond denote the condition for entering the if construct. Use (cond & flag) as the condition for entering the while construct, where flag is set to true, before entering the loop and set to false, immediately upon entry. This will guarantee that the statement inside the while construct is executed precisely once, thereby mimicking the if construct.

Task 2. The BNF rules for if-then-else statement are as follows:

< ifstmt >
  ! if < logicexpr > then < stmt >
  ! if < logicexpr > then < stmt > else < stmt >

Please illustrate the parse tree of the following sentential form.

if (done)
  if (!denom)
    number = 0;
  else number = sum/denom;

Is this grammar ambiguous? Please illustrate the parse tree of the sentential form. If the grammar is ambiguous, please draw two parse trees.

Solution: The student should show two different parse trees in his/her homework as same as page 149-149 in textbook. Note that the parse trees may be different.

Task 3. Prove that the following grammar is ambiguous:

< S >→ < A >
< A >→ < A > * < A > | < id >
< id >→ x | y | z

Solution: If the grammar generates a different parse tree, then we can say the grammar is ambiguous. The grammar generates two different parse trees. And the student should show two different parse trees in his/her homework.
Task 4. Given a grammar as follows:

\[
\begin{align*}
< \text{assign} > &\rightarrow < \text{id} > <= < \text{expr} > \\
< \text{id} > &\rightarrow A|B|C \\
< \text{expr} > &\rightarrow < \text{id} > + < \text{expr} > | < \text{id} > * < \text{expr} > | ( < \text{expr} > ) | < \text{id} >
\end{align*}
\]

Please show a parse tree for the following statement:

\[A = A * ( B * ( C + A ) )\]

Solution:

\[
< \text{assign} > \Rightarrow < \text{id} > = < \text{expr} >
\]

\[\Rightarrow A = < \text{expr} >
\]

\[\Rightarrow A = < \text{id} > * < \text{expr} >
\]

\[\Rightarrow A = A * < \text{expr} >
\]

\[\Rightarrow A = A * ( < \text{expr} > )
\]

\[\Rightarrow A = A * ( A * < \text{expr} > )
\]

\[\Rightarrow A = A * ( B * < \text{expr} > )
\]

\[\Rightarrow A = A * ( B * ( < \text{expr} > ) )
\]

\[\Rightarrow A = A * ( B * ( A * < \text{expr} > ) )
\]

\[\Rightarrow A = A * ( B * ( C + < \text{expr} > ) )
\]

\[\Rightarrow A = A * ( B * ( C + < \text{id} > ) )
\]

\[\Rightarrow A = A * ( B * ( C + A ) )
\]

Task 5. Perform the pairwise disjointness test for the following grammar rules.

a. \[< A > \rightarrow aB|b|cBB\]

b. \[< B > \rightarrow aB|bA|aBb\]

c. \[< C > \rightarrow aaA|b|caB\]

Solution:

1. FIRST(aB) = a, FIRST(b) = b, FIRST(cBB) = c, Passes the test

2. FIRST(aB) = a, FIRST(bA) = b, FIRST(aBb) = a, Fails the test

3. FIRST(aaA) = a, FIRST(b) = b, FIRST(caB) = c, Passes the test