Quiz

Compiler Construction, Fall 2010

Monday, December 20, 2010

Last Name : _____

First Name : _____

Exercise	**Points**	**Achieved Points**
1	10	
2	10	
Total	20	

General notes about this quiz

- This is an open book examination. You are allowed to use any written material. You are not allowed to use the notes of your neighbors.
- You have totally **three hours**.
- The points of the questions are not equal. It is advisable not to spend most of your time on the questions with less grade.

Problem 1: Lexical Analysis (10 points)

The increment operator in C++ is ++ (in fact C++ means incrementing the language C). Increment can be both prefix and suffix, so $++x$ and $x++$ effectively increase the value of x. Consider the alphabet $\Sigma = \{+, x\}$. We want to generate all the valid expressions that can be generated using these two symbols. The symbol + of the alphabet can be used in an increment operator ($x{+}{+}$), can be a binary operator ($x{+}x$) or a unary operator ($+x$). The lexical analyzer returns the following classes of tokens:

1. **PLUS:** The binary or unary operator +
2. **VAR:** The variable x
3. **INC:** The increment operator ++

For example consider the following expressions and their corresponding tokenizing.

Expression	Tokens	
$+x$	PLUS VAR	
$x++$	VAR INC	
$x+x+x$	VAR PLUS VAR PLUS VAR	
$++x++$	INC VAR INC	
$x++x$	VAR PLUS PLUS VAR	
$x+++x$	VAR INC PLUS VAR	
$x++++x$	VAR INC PLUS PLUS VAR	

a) Determine for each row of the table if the tokenizing can be done by a longest matching lexical analyzer or not. You can put a ✓ or × beside each row in the table.
For the negative answer justify why the result cannot be generated by a longest matching lexical analyzer.

b) Consider a restriction on the language which allows only three operands ++x, x++ and x and only one operator, binary +. Design a lexical analyzer which can tokenize the generated language by giving a deterministic finite automaton.

c) Do you need the longest match rule for the analyzer in part b?

Problem 2: Parsing (10 points)

Consider the following grammar on $\Sigma = \{\Rightarrow, \texttt{,}, \texttt{Int}\}$. The symbol \$ shows the end of file.

$$
\begin{array}{lll}
1: & S' & \rightarrow S\$ \\
2: & S & \rightarrow T \Rightarrow S \\
3: & T & \rightarrow S\texttt{,}T \\
4: & T & \rightarrow \texttt{Int} \\
5: & S & \rightarrow \texttt{Int}
\end{array}
$$

a) Compute the *first* and *follow* of the non-terminals S', S and T.

b) Determine if there is an input for which there exist at least two different parse trees.

c) Make an equivalent grammar with minimal changes so that the grammar can be recognized by an LL(1) parser.