HW 9 – Parse Trees, Ambiguous Grammars, LR and Recursive Descent Parsing

CS 421 – Spring 2014
Revision 1.0

Assigned Thursday, April 03, 2014
Due Sunday, April 14, 2013, 11:59 PM

1 Change Log

1.0 Initial Release.

2 Turn-In Procedure

This assignment is named `hw9`. Using your favorite tool(s), you should put your solution in a file named `hw9-solution.pdf`. Your answers to the following questions are to be submitted using the svn repository as described in the section `Instruction for Solving and Submitting Assignments` on the web-page: `http://courses.engr.illinois.edu/cs421/sp2014/mps/index.html`

3 Objectives and Background

The purpose of this HW is to test your understanding of

- How to create a parse tree for a given string with a given grammar
- How to disambiguate a grammar
- How to write a recursive descent parser for an LL(1) grammar

Another purpose of HW9 is to provide you with experience answering non-programming written questions of the kind you may experience on the second midterm and final.
Caution: It is strongly advised that you know how to do these problems before the second midterm.

4 Problems

1. (23 points) Consider the following grammar over the alphabet $\{if,\ then,\ else,\ +,\ x,\ y,\ z,\ (,\ )\}$:

$$
\begin{array}{lcl}
<\texttt{exp}> & ::= & <\texttt{var}> \mid \texttt{if} <\texttt{exp}> \texttt{then} <\texttt{exp}> \texttt{else} <\texttt{exp}> \mid <\texttt{exp}> + <\texttt{exp}> \mid (<\texttt{exp}>) \\
<\texttt{var}> & ::= & \texttt{x} \mid \texttt{y} \mid \texttt{z}
\end{array}
$$

   a. (9 points) Show that the above grammar is ambiguous by showing at least three distinct parse trees for the string "if x then x else x + y + z"

   b. (9 points) Write a new grammar accepting the same language that is unambiguous, and such that addtion $<\texttt{exp}> + <\texttt{exp}>$ has higher precedence than conditional $if\ <exp>\ then\ <exp>\ else\ <exp>$, and such that addition associates to the left.

c. (5 points) Give the parse tree for "`if x then x else x` + `y` + `z`" using the grammar you gave in the previous part of this problem.

2. (17 points) Given the following grammar over nonterminal `<m>`, `<e>` and `<t>`, and terminals `z`, `o`, `l`, `r`, `p` and `eof`, with start symbol `<m>`:

$$
\begin{aligned}
P0: &\ <\mathtt{m}>::=<\mathtt{e}>\mathtt{eof}\\
P1: &\ <\mathtt{e}>::=<\mathtt{t}>\\
P2: &\ <\mathtt{e}>::=<\mathtt{t}>\mathtt{p}<\mathtt{e}>\\
P3: &\ <\mathtt{t}>::=\mathtt{z}\\
P4: &\ <\mathtt{t}>::=\mathtt{o}\\
P5: &\ <\mathtt{t}>::=\mathtt{l}<\mathtt{e}>\mathtt{r}
\end{aligned}
$$

and Action and Goto tables generated by YACC for the above grammar:

State	Action							Goto		
	z	o	l	r	p	[eof]		<m>	<e>	<t>
st1	**s3**	**s4**	**s5**	**err**	**err**	**err**			**st2**	**st7**
st2	**err**	**err**	**err**	**err**	**err**	**a**				
st3	**r3**	**r3**	**r3**	**r3**	**r3**	**r3**				
st4	**r4**	**r4**	**r4**	**r4**	**r4**	**r4**				
st5	**s3**	**s4**	**s5**	**err**	**err**	**err**			**st8**	**st7**
st6	**err**	**err**	**err**	**err**	**err**	**a**				
st7	**err**	**err**	**err**	**r1**	**s9**	**r1**				
st8	**err**	**err**	**err**	**s10**	**err**	**err**				
st9	**s3**	**s4**	**s5**	**err**	**err**	**err**			**st11**	**st7**
st10	**r5**	**r5**	**r5**	**r5**	**r5**	**r5**				
st11	**r2**	**r2**	**r2**	**r2**	**r2**	**r2**				

where **st**i is state i, **s**i abbreviates **shift** i, **r**i abbreviates **reduce** i, **a** abbreviates **accept** and `[eof]` means we have reached the end of input, describe how the string `lzpor[eof]` would be parsed with an LR parser using these productions and tables by filling in the table on the next page. I have given you the first 5 cells in the first two rows to get you started. You will need to add more rows.

Stack	Current String	Action to be taken
Empty	`lzpor[eof]`	Initialize stack, go to state 1
st1	`lzpor[eof]`	