Memo: Tutorial 6

April 21, 2004

Question 1:

First(A) = $\{a\}$ and Follow(A) = $\{a, \$\}$
Thus the LL(1) parsing table is given by:

$M[N,T]$	a	$\$$
A	$A \to a\ A\ a$ $A \to \varepsilon$	$A \to \varepsilon$

Since we have two entries in $M(A, a)$, the grammar is not LL(1).

Question 2:

(a)

Grammar rule	Pass 1	Pass 2
stmt-sequence→*stmt stmt-seq*′		First(*stmt-sequence*)={**s**}
stmt-seq′→ ; *stmt-sequence*	First(*stmt-seq*′) = { ; }	
stmt-seq′→ε	First(*stmt-seq*′) = {;, ε}	
stmt→**s**	First(*stmt*)= {**s**}	

Thus
First(*stmt-sequence*)={**s**}
First(*stmt-seq*′) = {;, ε}
First(*stmt*)= {**s**}

(b)

Grammar rule	Pass 1
stmt-sequence→*stmt stmt-seq*′	Follow(*stmt-sequence*)={$} Follow(*stmt*)= {;, $} Follow(*stmt-seq*′) = {$}
stmt-seq′→ ; *stmt-sequence*	
stmt-seq′→ε	
stmt→**s**	

Thus
Follow(*stmt-sequence*)={$}
Follow(*stmt-seq*′) = {$}
Follow(*stmt*)= {;, $}

(c)

$M[N,T]$	s	;	$
stmt-sequence	*stmt-sequence*→*stmt stmt-seq*′		
stmt-seq′		*stmt-seq*′→ ; *stmt-sequence*	*stmt-seq*′→ε
stmt	*stmt*→**s**		

Question 3:

(a)

$M[N,T]$	(	**number**	)	+	−	∗	$
exp	*exp*→ *term exp*′	*exp*→ *term exp*′					
exp′			*exp*′→ε	*exp*′→ *addop term exp*′	*exp*′→ *addop term exp*′		*exp*′→ε
addop				*addop*→+	*addop*→−		
term	*term*→ *factor term*′	*term*→ *factor term*′					
term′			*term*′→ε	*term*′→ε	*term*′→ε	*term*′→ *mulop factor term*′	*term*′→ε
mulop						*mulop*→∗	
factor	*factor*→ (*exp*)	*factor*→ **number**					

(b)

Parsing stack	Input	Action
$ *exp*	**number** + **number** ∗ **number** − **number**$	*exp*→*term exp*′
$ *exp*′ *term*	**number** + **number** ∗ **number** − **number**$	*term*→*factor term*′
$ *exp*′ *term*′ *factor*	**number** + **number** ∗ **number** − **number**$	*factor*→ **number**
$ *exp*′ *term*′ **number**	**number** + **number** ∗ **number** − **number**$	match
$ *exp*′ *term*′	+ **number** ∗ **number** − **number**$	*term*′→ε
$ *exp*′	+ **number** ∗ **number** − **number**$	*exp*′→*addop term exp*′
$ *exp*′ *term addop*	+ **number** ∗ **number** − **number**$	*addop*→+
$ *exp*′ *term* +	+ **number** ∗ **number** − **number**$	match
$ *exp*′ *term*	**number** ∗ **number** − **number**$	*term*→*factor term*′
$ *exp*′ *term*′ *factor*	**number** ∗ **number** − **number**$	*factor*→ **number**
$ *exp*′ *term*′ **number**	**number** ∗ **number** − **number**$	match
$ *exp*′ *term*′	∗ **number** − **number**$	*term*′→*mulop factor term*′
$ *exp*′ *term*′ *factor mulop*	∗ **number** − **number**$	*mulop*→∗
$ *exp*′ *term*′ *factor* ∗	∗ **number** − **number**$	match
$ *exp*′ *term*′ *factor*	**number** − **number**$	*factor*→ **number**
$ *exp*′ *term*′ **number**	**number** − **number**$	match
$ *exp*′ *term*′	− **number**$	*term*→ε
$ *exp*′	− **number**$	*exp*′→*addop term exp*′
$ *exp*′ *term addop*	− **number**$	*addop*→−
$ *exp*′ *term* −	− **number**$	match
$ *exp*′ *term*	**number**$	*term*→*factor term*′
$ *exp*′ *term*′ *factor*	**number**$	*factor*→ **number**
$ *exp*′ *term*′ **number**	**number**$	match
$ *exp*′ *term*′	$	*term*′→ε
$ *exp*′	$	*exp*′→ε
$	$	accept

Question 4:

(a) and (b) A grammar can not be both LL(1) and ambiguous, since each string in an LL(1) grammar has exactly one leftmost derivation.
(c) No. The grammar in Question 1 is unambiguous but not LL(1).