CS331

COMPILER DESIGN

Spring, 2002

NAME:____

ANSWER ALL QUESTIONS ON THE EXAM SHEETS. IF YOU ARE UNSURE ABOUT THE MEANING OF A QUESTION, INDICATE THE ASSUMPTIONS YOU MAKE IN ANSWERING IT. APPROPRIATE CREDIT WILL BE GIVEN FOR REASONABLE ASSUMPTIONS. POINT VALUES FOR EACH QUESTION ARE GIVEN IN PARENTHESES.

Note: in all example grammars, the terminal symbols are given in boldface.

1. a. **(5)** Explain why the following grammar is not LL(*k*) for any *k*:

S → A | B
A → **aa**A | **aa**
B → **aa**B | **a**

b. **(5)** Explain why the following grammar is LL(1) but not SLR(1):

S → A**a**A**b** | B**b**B**a**
A → ε
B → ε

c. **(5)** Which of LL(1), SLR(1), and LR(1) can parse strings in the following grammar, and why?

E → A | B
A → **a** | **c**
B → **b** | **c**

2. **(10)** Consider the following two BNF grammars. In both cases:

- the terminals are { **A, B, C, D, E, F**, * },
- the non-terminals are { <expr>, <term>, <identifier> }, and
- the start symbol is <expr>.

For each grammar draw a parse tree for the expression

A B C * D * E F

If the grammar is ambiguous, state that it is ambiguous and give the reason for the ambiguity.

Grammar A:

<expr> ::= <term> | <expr> <term>
<term> ::= <identifier> | <identifier> * <term>
<identifier> ::= **A** | **B** | **C** | **D** | **E** | **F**

Grammar B:

<expr> ::= <term> | <term> <expr>
<term> ::= <identifier> | <term> * <term>
<identifier> ::= **A** | **B** | **C** | **D** | **E** | **F**

3. **(10)** Write the actions of an LR parse for the following string, for the grammar and parse table shown below:

aa1bbbb

Grammar:

(1) S → A
(2) S → B
(3) A → **a** A **b**
(4) A → **0**
(5) B → **a** B **b b**
(6) B → **1**

	ACTION					GOTO		
State	**a**	**b**	**0**	**1**	**$**	**S**	**A**	**B**
0	S1		S2	S3		11	4	5
1	S1		S2	S3			6	7
2		r4	r4		r4			
3	r6	r6	r6	r6	r6			
4	r1	r1	r1		r1			
5	r2	r2	r2	r2	r2			
6		S8						
7		S9						
8		r3						
9		S10						
10		r5			r5			
11					acc			

Stack	**Input**	**Action**

4. This question concerns the context-free grammar given below (capital letters denote non-terminals, small letters denote terminals, ε denotes the empty string, and S denotes the start non-terminal)

(1) S → W A B | A B C S
(2) A → B | W B
(3) B → ε | **y** B
(4) C → **z**
(5) W → **x**

a. **(10)** Fill in the following table with the First and Follow sets for the grammar (i.e., the First sets of the non-terminals, production right-hand sides, and suffixes of right-hand sides, and the Follow sets of the non-terminals).

X	First(X)	Follow(X)
S		
A		
B		
C		
W		
WAB		
ABCS		
WB		
yB		
BCS		
AB		
CS		

b. **(10)** Using the First and Follow sets in part (a), fill in the LL(1) parse table below:

	x	**y**	**z**	**EOF**
S				
A				
B				
C				
W				

c. **(5)** Is the grammar LL(1)? Justify your answer.

5. You are given the following grammar with the terminal symbols **(**, **)**, and **term** and non-terminals S, E and L.

(1) $S \rightarrow E\ \$$
(2) $E \rightarrow$ **term**
(3) $E \rightarrow$ **(** L **)**
(4) $L \rightarrow \varepsilon$
(5) $L \rightarrow E\ L$

a. **(5)** If the terminal **term** matches the character **x**, give *three* well formed strings in this grammar.

b. **(10)** A nearly complete collection of LR(0) sets of items for the above grammar is given below. Fill in the missing items and GOTO information. *Be sure you complete the entire set, including all necessary GOTOs.*

S_1: { $S \rightarrow \bullet\ E$
$E \rightarrow \bullet$ term
$E \rightarrow \bullet$ (L) }

S_2: GOTO(S_1,E)
{ $S \rightarrow E \bullet$ }

S_3: GOTO(S_1,**term**)
{ $E \rightarrow$ term$\bullet$ }

S_4: GOTO(S_1, **(**)
{ $E \rightarrow (\bullet$ L)
$L \rightarrow \bullet$
$L \rightarrow \bullet$ EL
$E \rightarrow \bullet$(L)
$E \rightarrow \bullet$ term }

GOTO(S_4, **(**) = S_4

S_5: GOTO(S_4,E)

S_6: GOTO(S_4,L)
{ $E \rightarrow$ (L$\bullet$) }

S_7: GOTO(S_6, **)**)

S_8:
{ $L \rightarrow$ EL$\bullet$ }

c. **(10)** Using the items in part (b), fill in the missing information for states 5 and 7 in the SLR(1) parse table. Note that the numbers in the reductions refer to the productions (1 – 5) as given above.

	(	**)**	**term**	**$**	**E**	**L**
1	Shift 4	error	Shift 3	error	2	
2	error	error	error	Accept		
3	Reduce 2	Reduce 2	Reduce 2	Reduce 2		
4	Shift 4 Reduce 4	Reduce 4	Shift 3 Reduce 4	Reduce 4	5	6
5						
6	error	Shift 7	error	error		
7						
8	Reduce 5	Reduce 5	Reduce 5	Reduce 5		

6. Consider the following grammar and associated semantic actions. In the actions, the operations And, Or, and Not are constructors for an abstract syntax tree data type.

$G \rightarrow F$	$G.p = F.p$
$F \rightarrow F_1 \wedge F_2$	$F.p = And(F_1.p, F_2.p)$
$F \rightarrow F_1 \vee F_2$	$F.p = Or(F_1.p, F_2.p)$
$F \rightarrow \neg F_1$	$F.p = Neg(F_1.p)$
$F \rightarrow F_1 \Rightarrow F_2$	$F.p = Or(Not(F_1.p), F_2.p)$
$F \rightarrow (F_1)$	$F.p = F_1.p$
$F \rightarrow id$	$F.p = id.lexeme$

a. **(5)** Say whether each attribute of a non-terminal is *inherited* or *synthesized* and why.

b. **(5)** Give the value of the attributes of *G* after parsing **¬ (A ∧ (A ⇒ B))**.

7. **(5)** Consider the following grammar:

$$S' \rightarrow S$$
$$S \rightarrow E = E$$
$$S \rightarrow \mathbf{i}$$
$$E \rightarrow E + \mathbf{i}$$
$$E \rightarrow \mathbf{i}$$

The start state (set of LR(0) items) for this grammar is as follows:

$$S_0 : \{\ S' \rightarrow \bullet S$$
$$S \rightarrow \bullet\ E = E$$
$$S \rightarrow \bullet\ \mathbf{i}$$
$$E \rightarrow \bullet\ E + \mathbf{i}$$
$$E \rightarrow \bullet\ \mathbf{i}\ \}$$

Also we have

$$S_3 = \text{GOTO}(S_0, \mathbf{i})$$
$$\{\ S \rightarrow \mathbf{i}\bullet$$
$$E \rightarrow \mathbf{i}\bullet\ \}$$

Because FOLLOW(S) = {\$} and FOLLOW(E) = {\$,+,=}, we have a REDUCE-REDUCE conflict for state 3 in the SLR(1) parse table.

Give the corresponding sets of LR(1) items for S_0 and S_3, which are used to construct the canonical LR parser, and explain why the conflict in the SLR(1) parse table is eliminated.