CMSC430 Spring 2011 Midterm 1 Solutions

1. (6 pts) Compiler front ends
   a. (3 pts) Explain how bottom-up parsing is different from top-down parsing.

      Bottom-up parsers transform the input to the start symbol by applying productions in reverse, whereas top-down parsers start from the start symbol.

   b. (3 pts) Explain how a LR parser can generate a right-most derivation even though it scans input from left to right

      Because it generates the derivation in reverse.

2. (32 pts) LL(1) parsing
   a. (12 pts) Compute FIRST and FOLLOW for S, A, B, and C for the following grammar, where ε is the empty string:

S →	AB	|	Cd
A →	a	|	Bf
B →	b	|	ε
C →	c	|	ε

	FIRST	FOLLOW
S	**a, b, c, d, f**	**$**
A	**a, b, f**	**b, $**
B	**b, ε**	**f, $**
C	**c, ε**	**d**

   b. (10 pts) Using the FIRST and FOLLOW information provided, construct the LL(1) parse table for the following grammar.

S →	aA	|	A
A →	bA	|	ε

	FIRST	FOLLOW
S	a, b, ε	$
A	b, ε	$

	a	**b**	**$**
S	**S → aA**	**S → A**	**S → A**
A		**A → bA**	**A → ε**

   c. (2 pts) Is the grammar in part 2(b) LL(1)? Explain.

      Yes, because there are no conflicts in the LL(1) parse table.

   d. (8 pts) Using the following LL(1) parse table, parse the input "ba". You only need to show the stack and remaining input at each stage of the parse.

	a	b	$
S	S → ba	S → A	S → C
A	A → ba	A → bBa	A → ε
B	B → ε	B → a	B → ε
C	C → ab	C → ba	C → ε

Stack	**Input**
$ S	**ba$**
$ A	**ba$**
$ a B b	**ba$**
$ a B	**a$**
$ a	**a$**
$	**$**

3. (12 pts) Shift-reduce parsing

Given the following ACTION/GOTO table, show the parse if the current stack contents are 0 b 4 B 1 (i.e., current state is 1) and the remaining input is "d$".

State	ACTION			GOTO	
	d	e	$	S	B
0	Shift 2	Reduce S → Bb	Reduce S → bB	2	1
1	Shift 3	Reduce B → d	Accept	3	4
2	Shift 1	Shift 4	Accept	2	4
3	Reduce S → Bd	Reduce B → d	Reduce S → bBd	1	2
4	Reduce B → e	Shift 3	Reduce S → bSe	4	3

Stack	**Input**	**Action**
0 b 4 B 1	**d $**	**Shift 3**
0 b 4 B 1 d 3	**$**	**Reduce S → bBd**
0 S	**$**	**Goto 2**
0 S 2	**$**	**Accept**

4. (20 pts) LR(1) canonical sets of items

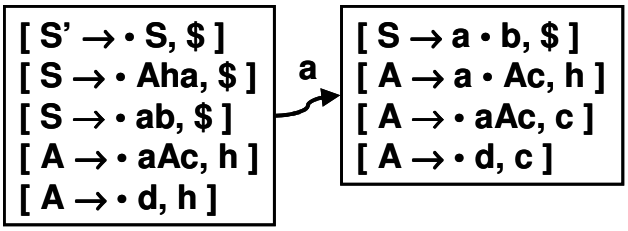
Given the following augmented grammar, derive the following two states in its canonical sets of LR(1) items: the starting state, and the state reached on a transition for shifting "a" from the starting state.

S' → S

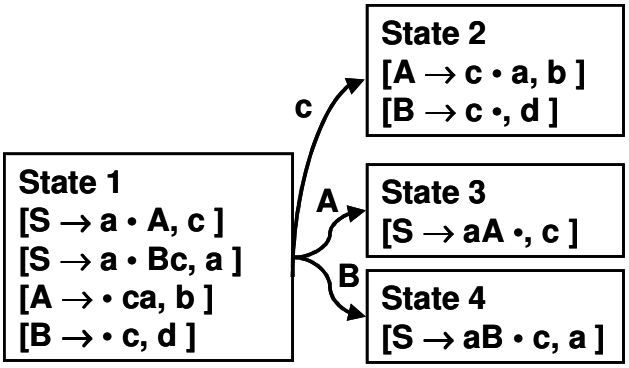
S → Aha | ab

A → aAc | d

B → b



5. (12 pts) LR(1) parse table construction
   Given the following sets of LR(1) items, construct the corresponding entries in the LR(1) parse table.



State	ACTION				GOTO	
	a	b	c	d	A	B
1			Shift 2		3	4
2				Reduce B → c		
3			Reduce S → aA			
4						

6. (12 pts) LR(1) conflicts
   a. (8 pts) Consider the following LR(1) items in a single state of a shift/reduce parser. List any conflicts that exist and describe for what lookaheads they occur.

Shift/reduce conflict:
[E → ca • c, a] and [H → Ha • , c] for lookahead c

Reduce/reduce conflict:
[F → aAa •, a] and [G → ba • , a] for lookahead a

[A → a • bAb, d]
[B → a • bc, c]
[C → da • da, a]
[D → aa • d, c]
[E → ca • c, a]
[F → aAa • , a]
[G → ba • , a]
[H → Ha • , c]

   b. (4 pts) Describe how compilers may intentionally use shift-reduce parsers containing conflicts.

Because it can resolve the shift/reduce conflicts in a designated way to implement operator precedence and associativity.

7. (6 pts) SLR(1) & LALR(1) parsers
   a. (3 pts) What is the main advantage of SLR(1) and LALR(1) parsers, as compared to LR(1) parsers?

Smaller parse table (and thus less memory).

   b. (3 pts) What is the main disadvantage of SLR(1) and LALR(1) parsers, as compared to LR(1) parsers?

Unable to handle as many grammars (i.e., may have conflicts not found in LR(1) parser).