CS143 Handout 23
Winter 2006 February 14, 2006

CS143 Midterm Exam

This is an open-note exam. You can refer to any course handouts, handwritten lecture notes, and printouts of any code relevant to a CS143 assignment. You may **not** use any laptops, cell phones, or handheld devices of any sort. The exam is written to take about 90 minutes, but you can take up to three hours.

SCPD students should fax their exam back to 650-723-6092 when finished, and then have your on-site courier deliver the original back to us. Feel free to telephone me 415-205-2242 if you have any questions.

Good luck!

`leland` username: _____

Last Name: _____

First Name: _____

SUID: _____

I accept the letter and spirit of the honor code. I've neither given nor received aid on this exam. I promise to write as neatly as I'm capable.

(signed) _____

		Score	Grader
1. DFAs and CFGs	(10)	_____	_____
2. LL(1) Expressions	(15)	_____	_____
3. LR(0) versus SLR(1)	(15)	_____	_____
Total	**(40)**	_____	_____

Problem 1: Finite Automata and Regular Grammars [10 points]

a.) [5 points] Draw a DFA that accepts all strings in (a + b)* that do **not** contain bababb as a substring.

b.) [5 points] Present a context-free grammar that generates the language accepted by your DFA from part a. [Hint: The number of nonterminals in your grammar should be roughly equal to the number of states in your DFA.]

Problem 2: LL(1) Expressions [15 points]

Consider the following grammar for expressions, consisting of array accesses, addition, and assignments:

$$E \rightarrow E[E]$$
$$E \rightarrow E + E$$
$$E \rightarrow E = E$$
$$E \rightarrow (E)$$
$$E \rightarrow \text{id}$$

This grammar is ambiguous. We'd like to write an unambiguous grammar for the same language such that array accesses have higher precedence that assignments, and both array accesses and assignments have higher precedence than addition. Addition is left-associative, and assignment is right-associative.

a.) [7 points] Write an LL(1) grammar which accepts the same language and has the desired operator precedence and associativity. You'll first need to install some additional nonterminals to support the precedence. You'll then need to transform the resulting grammar to be LL(1).

b.) [8 points] Build the LL(1) parsing table for your LL(1) grammar from part a.

Problem 3: LR(0) versus SLR(1) [15 points]

Consider the already augmented grammar:

$$S' \rightarrow S$$
$$S \rightarrow Sd$$
$$S \rightarrow cAe$$
$$A \rightarrow Sa$$
$$A \rightarrow S$$
$$A \rightarrow a$$

Build the family of LR(0) configurating sets for this grammar in goto-graph form. Draw arrows from each set to its successors, and label each arrow with the appropriate grammar symbol. Note which states have conflicts, identify the type of conflict, and note whether an upgrade to SLR(1) would resolve the conflict held by that state.