CS 4300, Fall 2009 – Compiler Theory
Exam
December 4, 2009

- All answers must be your own work. If you consult any sources besides your course notes and textbook, you must cite them. You may not discuss this exam with other students.
- To receive full credit, **show your work** and **write legibly**.
- Write each problem on a separate sheet of paper.
- This exam is due on December 18, 2009 at 2:00 pm.

1. Use **flex** to implement a lexical analyzer that recognizes the following 7 tokens: identifiers, 2 keywords, and 4 operators.

   identifiers → [A-Z] ([A-Z] | [a-z])*
   keywords → "if" | "while"
   operators → "+" | "*" | "=" | ";"

You only need to return token types. No lexical error recovery is needed. No main function is needed.

2. Consider the following grammar

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

   (a) Compute the FIRST set for each of the nonterminal S, A, and B.
   (b) Compute the FOLLOW set for each of the nonterminal S, A, and B.
   (c) Write a recursive decent parser for this grammar. You may assume that functions match() and error() are already available.
   (d) Construct a nonrecursive LL(1) parsing table for this grammar. No error recovery is needed.

3. Consider the following grammar

   S → a *A* d | a c e | b *A* e
   A → c

(a) Construct the SLR(1) parsing table for this grammar. Is this grammar SLR(1)?
(b) Construct the LR(1) parsing table for this grammar. Is this grammar LR(1)?
(c) Construct the LALR(1) parsing table for this grammar. Is this grammar LALR(1)?

4. Consider the following productions in the grammar for language **Lotus**

   function_definition → **int Identifier (** *parameters* **)** *function_body*
   parameters → **empty** | *parameter_list*
   parameter_list → **int Identifier** | *parameter_list* **, int Identifier**

Give Bison semantic rules for these productions to construct abstract syntax trees for function definitions. You can use the following routines for constructing nodes of an abstract syntax tree

   Tree build_function(char * name, Tree params, Tree body);
   Tree build_list(Tree front, Tree last);
   Tree build_id(char * var);