CSC 488/2107 Language Processors **Winter 2007**
Midterm Test [15% of final mark] **March 1, 2007**

Total marks 50 - Total time is 50 minutes. Answer all 4 questions.
Instructions: This midterm is open book, open notes. Non-programmable calculators allowed. No electronic communication devices allowed.
The line numbers in grammars are for reference only and are not part of the grammars. Ellipses … indicate omitted, correct text. ε indicates an empty string.
If you need to make any assumptions in order to answer a question, state the assumptions clearly in your answer book.
Remember: an explanation of your reasoning is always more important than the (correct) Yes/No answer.

I. [10 marks] Regular expressions and automata.

1. **[4 marks]** Write a regular expression defining the language of comments consisting of a string surrounded by /* and */, unless it is inside double quotes (”). For example, /* aaa */ is a comment and so is /* a”*/”a */, but /* aa”*/ is not. Assume your alphabet is Σ.

   $/* \left(\Sigma - \{”\} \mid ” \, (\Sigma - \{”\})^* \, ” \right)^* \, */$

   Any correct solution receives full marks. Any reasonable attempt at a solution receives two marks.

2. **[4 marks]** Design a finite automaton for this language.

   The easiest solution is to construct a finite state machine from the regular expression used above. Any correct solution receives full marks. Any reasonable attempt at a solution receives two marks.

3. **[2 marks]** Is your automaton deterministic or non-deterministic? Explain your answer.

   An automaton is deterministic iff. for each state there is exactly one out-going edge per symbol and there are no epsilon transitions.

   Full marks for a correct answer; no marks for anything else.

II. [16 marks] LL grammars and parsing.
Consider the following context-free grammar:

1: S ::= $S\,(S)\,S$
2: | ε

1. **[2 marks]** Give either a leftmost or a rightmost derivation for string (()()). Explain which one you produced.

   A leftmost derivation of (()()) is:

```
-> (S) S
-> (S (S) S) S
-> ((S) S) S
-> (() S) S
-> (() S (S) S) S
-> (() (S) S) S
-> (() () S) S
-> (() ()) S
-> (() ())
```

A rightmost derivation would reduce the rightmost non-terminal first at each step.

Two marks for a correct answer; zero marks otherwise.

2. **[3 marks]** Is this grammar ambiguous? Justify your answer.

   Yes, because there is more than one leftmost derivation of the string above.

   Three marks for a correct answer, one mark for a correct answer without a justification.

3. **[3 marks]** What is the language generated by this grammar?

   Zero or more arbitrarily nested sequences of paired parentheses.

   Three marks for a correct answer, one mark for an imprecise answer.

4. **[4 marks]** Is this grammar LL(1)? If not, construct an equivalent one that is.

   (1) No. A correct response could be either (1) that it is ambiguous or (2) that it contains left recursion.

   Any correct, justified answer receives one mark; zero marks otherwise.

   (2)

   ```
   S -> (S) S | epsilon
   ```

   Any correct answer gets three marks; a reasonable attempt gets one mark.

5. **[4 marks]** Prove that it is LL(1).

   The director sets for the above grammar are as follows:

   ```
   S -> (S) S   { ( }
   S -> epsilon ( ), $ }
   ```

   Since the sets are disjoint, the grammar is LL(1).

   Any correct answer, with justification, gets four marks; an reasonable attempt gets one mark.

1. **[8 marks]** Show that the following grammar is SLR(1):

   1: S ::= $S\,A$
   2: | A
   3: A ::= a

   Step 1. The LR(0) items for this grammar are as follows:

```
I0:
S' -> .S
S  -> .S A
S  -> .A
A  -> .a

I0 -a-> I1:
A  -> a.

I0 -A-> I2:
S  -> A.

I0 -S-> I3:
S' -> S.
S  -> S .A
A  -> .a

I3 -a-> I1

I3 -A-> I4
S  -> S A.
```

   Step 2. There are no shift-reduce conflicts in the parsing table derived from these items, so the grammar is SLR (1).

   Eight marks for a correct answer with justification; five marks for correct LR(0) items.

2. **[3 marks]** Without building the table for LALR(1), explain why this grammar is LALR(1) as well. Just saying that $L(\text{SLR}(1)) \subseteq L(\text{LALR}(1))$ gets 0 marks.

   Since the grammar is SLR, this means that follow sets are enough to tell the states apart. With an LALR, there will be the same core sets, so no more states will be reachable. Furthermore, the decisions will be made based on lookahead sets which are subsets of follow sets. Therefore, if there were no shift/reduce conflicts in SLR(1), there will not be any in LALR(1).

   Three marks for a correct answer; one mark for a partial answer.

There is only one element on the stack for LL and you have to make predictions only based on that. In LR, you are collecting the entire handle, so there is more information on which to base a decision.

Three marks for a correct answer; one mark for a partial answer.

IV. [10 marks] Run-time analysis.
Show the *run-time* checks that have to be make to make sure that the following program does not have type and range errors.

```
var i, j:  2..50
    m, n: integer
    A: array [3..75] of -10..-5

i = i-1;
  result of i - 1 is in 2 .. 50
m = i+j;
A[m] = n;
  n is in -10 .. -5
  m is in 3 .. 75
A[3*m-7] = -2 * j;
  -2 * j is in -10 .. -5
  3 * m - 7 is in 3 .. 75
```

Two marks for each correct check. Note that the question specifically asks only for necessary *run-time* checks. Checking that, for example, `m` is of type integer can and should be done at *compile time*.