University of Toronto

CSC 488/2107 Language Processors **Winter 2005**
Midterm Test [15% of final mark] **Feb. 24, 2005**

Total marks 90 - Total time is 50 minutes. Answer all 6 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.

1 [15 marks + bonus]. Languages and automata.

1. **[12 marks]** Create an automaton recognizing a language over an alphabet $\{0, 1\}$ where the number of 0's is odd AND the number of 1's is odd.
2. **[3 marks]** Is this language regular? I.e., can it e described by a regular expression?
3. **[Bonus: 6 marks.]** If you answered yes to the previous question, give the regular expression for this language.

2 [15 marks]. Construct a DFA for a language

1: S $\longrightarrow$ B C D D
2: B $\longrightarrow$ D B | ϵ
3: C $\longrightarrow$ '+'
4: D $\longrightarrow$ '+' | '−'

3 [20 marks]. Show that the following grammar is LR(1) but not LALR(1):

1: S $\longrightarrow$ p X q
2: $\longrightarrow$ X r
3: $\longrightarrow$ p Y r
4: $\longrightarrow$ Y q
5: X $\longrightarrow$ w
6: Y $\longrightarrow$ w

4 [15 marks]. What errors may occur in the following piece of code? (Here we are using a C-like language but with nested method scopes.) Describe which semantic checks will catch these errors. Separate your list into static and run-time checks.

```
main() {
   ...
   void mine () {
     ...
     int A = B[i+3] * C - compute(B,C);
```

5 [10 marks]. Let the following Pascal type declaration be given

```
type  link = ↑ cell;
      cell = record
           info : integer;
           next : link
      end;
```

Here, link is a pointer to cell, and its type is *pointer*(cell). Type of a tuple A : T is indicated as A × T, and type of a record with fields B and C is indicated as *record*(B × C). Listed below are six type expressions. Which of these are name equivalent and which are structure equivalent? For your answer, you can give a 6 × 6 table, indicating each entry as S (structure), N(name) or blank (neither).

1. link
2. *pointer*(link)
3. cell
4. *pointer*(cell)
5. *record* ((info × integer) × (next × *pointer*(cell))
6. *pointer*(*record* ((info × integer) × (next × *pointer*(cell))))
7. *pointer*(*record* ((info × integer) × (next × link)))

6 [15 marks]. Suppose we are working with an n-dimensional array A[0..u_1, 0..u_2, ..., 0..u_n]. A is *sparse*: non-zero entries are allowed only on the diagonal. Show how efficient array storage for such sparse arrays can be laid out by the compiler. How much space would be necessary? What is the general case of an array subscript calculation, i.e., how a reference to an array element A[E_1, E_2, ..., E_n] is transformed into a memory address of this array element?