University of Toronto

CSC488S/CSC2107S Compilers & Interpreters **Winter 2002/2003**
Midterm Test [15% of final mark] **March 4, 2003**

Total marks 100 - Total time is 50 minutes. Two pages, 5 questions. Answer all questions.
Instructions: This midterm is open book, open notes. Non-programmable calculators allowed.
No electronic communication devices allowed.
The line numbers in example programs and grammars are for reference only and are not a part of the programs or grammars.

1. [20 marks] The representation of information is evolving toward the use of meta languages like XML as a standard encoding format. Suppose that this evolution has gone forward so that even text files containing source programs are now encoded in XML. For example, the program fragment:

```
if ( X != 13 ) {
   Y = 12 ;
   Z = X -1 ;
}
else
   X++ ;
```

Might be encoded as

```
<RESERVED>if</RESERVED> <PARENTHESIS> <IDENTIFIER>X</IDENTIFIER>
<OPERATOR>!=</OPERATOR> <INTEGER>13</INTEGER> </PARENTHESIS>
<BRACKET> <IDENTIFIER>Y</IDENTIFIER> <OPERATOR>=</OPERATOR>
<INTEGER>12</INTEGER> <OPERATOR>;</OPERATOR> <IDENTIFIER>Z</IDENTIFIER>
<OPERATOR>=</OPERATOR> <IDENTIFIER>X</IDENTIFIER> <OPERATOR>-</OPERATOR>
<INTEGER>1</INTEGER> <OPERATOR>;</OPERATOR> </BRACKET>
<RESERVED>else</RESERVED> <IDENTIFIER>X</IDENTIFIER>
<OPERATOR>++</OPERATOR> <OPERATOR>;</OPERATOR>
```

Describe how you would design a compiler scanner to process source programs that are encoded in this way.
Emphasize what would be *different* from the traditional scanner that was discussed in lecture.

2. [20 marks] Each of the grammars listed below contains one or more errors that prevents it from being processed by an LR(1) parser generator. For each grammar, describe the error(s), and give a revised grammar that corrects the problem.

Grammar A				Grammar B				Grammar C			
1	S	→	A S	1	S	→	A S	1	S	→	A b S
2		→	A S B S	2		→	c	2		→	B a S
3		→	c	3	A	→	A a	3	A	→	A a
4	A	→	a	4		→	D a	4		→	b
5	B	→	b	5	D	→	a	5	B	→	b A a

3. [25 marks] Describe semantic analysis checks that a Java compiler should perform on the program fragments listed below

Fragment A

```
void method() {
      int I = 0 ;
      while( I < 10 ) {
            int J = 0 ;
            I ++ ;
      }
      System.out.println( I ) ;
}
```

Fragment B

```
for( int K = 0 ; K < data.length ; K++ ) {
      if( data[ K ] == target ) {
            index = K ;
            break ;
      }
}
```

4. [20 marks] Numerical analysts often take advantage of symmetries in a problem to make more efficient use of memory to store large amounts of data. One well known technique is the use of upper triangular storage for symmetric matrices.

A matrix $A[N,N]$ is *symmetric* if and only if $\forall I,J \quad A[I,J] = A[J,I]$
Since the information above and below the diagonal is the same, it is only necessary to store the diagonal and the upper triangular region above the diagonal.

Assume you are implementing a Numerical Analyst friendly extension to C in which upper triangular matrices are built in data types, e.g. a programmer might declare:
doubleupper A[100][100] ;

Design an implementation of upper triangular matrices
a) show how the matrices are laid out in memory.
b) based on your memory layout, give an algorithm for calculating the address of an arbitrary array element $B[E_1][E_2]$ where B is an MxM doubleupper matrix and E_1 and E_2 are integer expressions.

5.[15 marks] The symbol table algorithms described in lecture were developed for traditional procedural programming languages like Algol-60, Pascal and C. Modern Object Oriented programming languages have changed some of the fundamental design assumptions on which these symbol table algorithms were based. In particular:

- Since each object introduces it's own scope of declaration, there are many more scopes than in traditional languages.
- The proliferation of scopes means that the compiler will have to deal with many more declared identifiers that was typical in traditional languages.
- Inheritance between classes, especially multiple inheritance has made name resolution (locating the correct declaration for a given identifier) much more difficult.

Discuss how the traditional symbol table algorithms might be modified to better accommodate the revised design assumptions