CS4905 Introduction to Compiler Construction
Assignment 1 January 11, 2007, due January 23, 2007

1. Give a brief definition of the following terms: compiler, lexical analyzer, semantic analysis, symbol table, context free grammar, parse tree, intermediate representation, code emission, computer language.

2. Textbook exercises on writing regular expressions, 2.1 b., 2.1 c., 2.1 d., 2.1 f, 2.1 g on p.34.

3. In some programming languages, identifiers can have embedded underscore characters. The first character cannot be an underscore, two underscore characters cannot appear in succession and an identifier cannot end with an underscore. For the set of strings representing these identifiers,
(a) give a regular expression describing this set of strings,
(b) give a deterministic finite automaton (DFA) to recognize these identifiers.

4. Textbook exercises on finite state automata, 2.3 a., 2.3 b., 2.3 c., 2.4 a., 2.4 b., 2.5 a., 2.5 c. on p.35.

5. Consider the following context free grammar:

$S \rightarrow$ (S) S | $\in$

(a) Show how the string (()(())) can be generated by this grammar (i.e. show a derivation).
(b) Construct a parse tree for the string in (a).
(c) What language is generated by this grammar?
(d) What differences arise if the grammar is changed to

$S \rightarrow S$ (S) | $\in$

?

6. Formulate a context-free grammar for the language of parenthesized logical expressions consisting of the logical variable b and the logical operators $\neg$ (not), $\vee$ (or) and $\wedge$ (and), such that $\neg$ has higher priority than $\wedge$, which in turn has higher priority than $\vee$. Give a derivation and associated parse trees for the following sentences:

(i) $\neg$ b $\vee$ b $\wedge$ b
(ii) $\neg$ (b $\wedge$ b) $\vee$ b
(iii) (b $\wedge$ b) $\wedge \neg$ b

7. Implement a JavaCC (Java compiler compiler) program to recognize the tokens given in the text in Figure 2.2 on p.20 (note the JavaCC specification given on p.31 of the text).
(a) Run your "lexer" program generated by javacc using the following input data:

```
if if5
1.6
Time1
r u
--commenta
-2.45
4.
21R
7 .0
8
-2.4E-3
-- commentb
```

```
if a < b then c = -6
if t > 5 then t = 0.4E-2
Done
```

Each time your lexer recognizes one of the tokens properly, have it print a statement saying e.g.

```
1.6: I recognize REAL
```

if a `REAL` token is recognized for the lexeme `"1.6"`. Print the lexeme before the "I recognize ..." statement by including a `Token` object in the Input specification. For example, the statement to recognize a `REAL` token might look something like the following:

```
t = <REAL> { System.out.println(t.image + ": I recognize REAL " ); }
```

where `t` is of type `Token`.

Make sure you hand in your complete program listing (i.e. the `.jj` file and the `Main.java` file) for this question, along with your output. Instructions on how to get started with lexical analysis using JavaCC are covered in Lab 1.

(b) Alter the regular expressions to allow the lexer to recognize negative real numbers (i.e. those preceded by a minus sign), real numbers with exponents, comments preceeded by `"--"` and blanks, LESS (`"<"`), GREATER (`">"`), THEN (`"then"`) and ASSIGN (`"="`) tokens, and run your program on the above data once more.

(c) Implement a lex program to recognize the same tokens as your JavaCC program of 7. (b), including the negative real numbers, and test it with the data in 7. (a).

Please put everything on 8.5 by 11 inch paper, typed (except for equations and figures), stapled upper left, with all pages numbered (numbering by hand is OK). Include a title page with your name, student ID number, date, course title, assignment number and table of contents (if appropriate). 0.5 marks will be deducted if this format is not followed.