Homework 1
Given: 1/22/09
Due Date: 2/3/09

1. Consider the following grammar $G = (N, T, S, P)$, where the set of non-terminal symbols $N = \{\text{prog}, \text{id\_list}, \text{decls}, \text{comp\_stmts}, \text{type}, \text{stmt\_list}, \text{stmt}\}$; the set of terminal symbols $T = \{\text{prog, id, left\_paren, right\_paren, semi\_coln, period, comma, var, coln, int, real, begin, end, s}\}$; the start symbol is $S = \text{prog}$; and the set of production rules $P$ are as follows:

$$
\text{prog} \rightarrow \text{prog id left\_paren id\_list right\_paren semi\_coln decls comp\_stmts period}
$$

$$
\text{id\_list} \rightarrow \text{id} \mid \text{id\_list comma id}
$$

$$
\text{decls} \rightarrow \text{decls var id\_list coln type semi\_coln} \mid \epsilon
$$

$$
\text{type} \rightarrow \text{int} \mid \text{real}
$$

$$
\text{comp\_stmts} \rightarrow \text{begin stmt\_list end}
$$

$$
\text{stmt\_list} \rightarrow \text{stmt} \mid \text{stmt\_list semi\_coln stmt}
$$

$$
\text{stmt} \rightarrow \text{s}
$$

(a) Construct a grammar $G_1$ by remove all left-recursions from $G$ such that $G_1$ is equivalent to $G$.

(b) Construct the FIRST and FOLLOW sets for each non-terminal symbol in $G_1$.

(c) Construct a predictive parser (i.e., a parsing table) for $G_1$.

(d) Apply your parser to the following input string and show all stack operations.

```
prog id left\_paren id\_list comma id right\_paren semi\_coln
var id coln int semi\_coln
var id comma id coln real semi\_coln
begin
s semi\_coln s
end period
```

2. The following grammar called *Equation Typesetting Preprocessor (EQN)* was made by Kernighan and Cherry [1975].

$$
(0) \quad E' \rightarrow E
$$
where the set of non-terminal symbols is $N = \{ E', E \}$ and the set of terminal symbols is $\{ \text{sub}, \text{sup}, \{ \}, c \}$. Note that the above grammar is ambiguous for several reasons. The grammar does not specify the associativity and precedence of the operators sub and sup. However, many typesetters would prefer to typeset an expression like $a \text{ sub } i \text{ sup } 2$ as $a_i^2$ rather than as $a_i^2$. Let’s try to construct an LL(1) parser, and identify any problems.

(a) Construct a grammar $G_1$ by removing all left-recursions from $G$ such that $G_1$ is left-factored and equivalent to $G$.

(b) Construct the FIRST and FOLLOW sets for each non-terminal symbol in $G_1$.

(c) Show that $G_1$ is not an $LL(1)$ grammar by constructing a predictive parsing table. Discuss about the problems you encounter in the construction of a parsing table and how you would solve such problems.