In this project you will write a lexical analyzer and a parser for a programming language [no translation yet]. The following grammar defines (a variation of) the programming language PL/0. It supports unlimited nested procedure calls, uses only one variable type (integer) as well as constants. The grammar definition of the language is as follows:

Grammar

Tokens: relation = {,<,>,<,>,<,>=,=}. ident: any sequence of letters and digits that starts with a letter. number: any sequence of digits (interpreted as type integer). reserved words begin, call, do, end, if, procedure, then, var, while, writeln. Punctuation marks . , ; := = + - * / ( )

<program>   →  <block> .
<block>      →  <const_decl> <vars_decl> <procs_decl> <statement>
<const_decl> →  const <seqass> ;
               | ϵ
<seqass>    →  ident = number
               | ident = number , <seqass>
<var_decl>  →  var <seqident> ;
               | ϵ
<seqident>  →  ident
               | ident , seqident
<proc_decl> →  <proc_decl> procedure ident ; <block> ;
               | ϵ
<statement> →  ident := <expression>
               | call ident
               | begin <seqstmt> end
               | while <condition> do <statement>
               | if <condition> then <statement>
               | writeln ( <expression> )
               | ϵ
<seqstmt>   →  <statement>
               | <statement> ; <seqstmt>
<condition> →  <expression> relation <expression>
<expression> →  <expression> + <expression>
              | <expression> - <expression>
              | <expression> * <expression>
              | <expression> / <expression>
              | ident
              | number
              | ( <expression> )

Note that expressions are defined in an ambiguous way. You should disambiguate the
grammar by employing the specification of precedence rules within yacc.

Your lexical analyzer should insert into a symbol table an entry for each token that it discovers. In particular you should declare the following data structure for your symbol table. Note that for simplicity we will use a static size symbol table. The declaration is below:

```c
/* ##########################################
  definition of an entry to the symbol table
  type = 0 when the identifier is a constant,
  type = 1 for variables, 
  type = 2 for procedure names. 
  line_no = is the line number the identifier is declared.
  ########################################## */

typedef struct {
  char name[MAX_ID_LEN];
  int type;
  int line_no;
} entry;

entry table[MAX_TABLE_LEN];

/* ##########################################
  declaration of the install_id procedure
  ########################################## */

entry *install_id(char *);

Note that the same identifier maybe declared twice when e.g., inside a procedure and side of a procedure. Because of this some special care will be needed so that you will enter the appropriate values into the symbol table. Also, note that procedures take no parameters (they will inherit variables of the parent procedures or main program). The scope of a variable is defined in a standard way (i.e., a procedure will inherit a variable that was declared in a higher level). Your Yacc program may also output arbitrary statistics of your liking.

Error-Handling

Your Yacc program should also handle the following errors:

- Syntax Errors. It should print the line of the error and recover (i.e., continue parsing).
- It should report the use of an identifier without having declared it first.
- Report type mismatch errors (e.g., when a procedure identifier is used in an expression or when a constant identifier is used to the left of an assignment).
- Report if an identifier is declared twice.
What to Submit

Submit (i) short description of your code (1 page) (ii) the LEX and YACC code (iii) the output of your program in the four sample files to be found in the web-site of the class – look into the announcement section a few days before the due date. (note: if you are using windows you can present your output by pressing “PRINTSCREEN” and copying / pasting to a word-document).

The whole assignment should be prepared as a single file using MS Word or other word-processing / type-setting software package of your liking. The first page should contain only: (i) your name(s), (ii) the class number “CSE 244”, (iii) the semester “Fall 2004”, (iv) the project number “Project #2”.