Programming Languages
Assignment 3: A Scheme Parsing Program
(Due: 4/28/15)

Consider the grammar G = (S, N, P, Σ) where

S = (Program)
N = (statement_list, statement, expr, symbol, op)
Σ = (if, bool, then, while, id, const, =, +, -, *, /)

P = (

Program	→	statement_list
statement_list	→	statement statement_list
	→	statement
statement	→	if bool then statement_list
	→	while bool statement_list
	→	id = expr
expr	→	symbol op symbol
symbol	→	id
	→	const
op	→	+ | - | * | /

You are to write a Scheme program that correctly parses all valid programs in L(G), the language defined by G. The parsing activity will conform to the syntactic structure defined by L(G). *The Scheme program will report (a) the total number of statements in the program, and (b) the maximum nested depth for a program.* You can assume that you will be given only valid programs. Each program will be provided as a parenthesized list of statements, each of which is included in its own parentheses.

The Program:

```
if bool then
   while bool
         id = id + const
         if bool then
                  id = const / const
id = const * id
```

Will Be Provided As:

```
( ( if bool then
      ( while bool
           ( id = id + const )
           ( if bool then
                 ( id = const / const)
           )
      )
   )
   ( id = const * id )
)
```

Note: The nesting of parentheses is used to indicate subordinate (or block) statement(s).

Some sample programs and expected outputs are provided below:

((id = id - const))

(NumberOfStatements: 1 MaximumDepth: 0)

((id = id + id) (id = id - id))

(NumberOfStatements: 2 MaximumDepth: 0)

((if bool then (id = const / const)))

(NumberOfStatements: 2 MaximumDepth: 1)

((if bool then (id = const / const)(id = id + id)))

(NumberOfStatements: 3 MaximumDepth: 1)

((if bool then (id = const / const))(while bool (id = id - const)))

(NumberOfStatements: 4 MaximumDepth: 1)

((id = id + id)(if bool then (id = const / const)(id = id + id))(while bool (id = id - const)(id = id - id)))

(NumberOfStatements: 7 MaximumDepth: 1)

((id = id + id)(if bool then (if bool then (id = id + id))(id = const / const)(id = id + id))(while bool (id = id - const)(id = id - id)))

(NumberOfStatements: 9 MaximumDepth: 2)

Be sure that you understand how the list structure reflects the program structure (statements & nesting) BEFORE attempting to write a Scheme program.

Your program must be named "parser" and take only one argument. I will add tests at the end of the file containing your parser. The tests will be of the form "(parser '(-----))"

I will then execute your program on "rlogin" using "plt-r5rs < file_name" where file_name is the name of the file containing your scheme program, and plt-r5rs (/usr/local/bin/plt-r5rs) is the UNIX scheme interpreter provided by Racket.

If you download the Scheme interpreter for windows, *the Language you will select will be "R5RS" under Legacy Languages*. This will define how your interpreter works and to which functions you have access. **Note, however, that I will execute you programs on Rlogin!**

You are restricted to:

The LISP Numeric Primitives: +, -, *, /

The LISP Predicates: =, <, >, >=, <=, <>, even?, odd? and zero?

symbol, number, list? and null?

The LISP functions: define, quote (‘), car, cdr, cons, cond, eq?, and equal?

The car and cdr family of functions, e.g., caar, cadr, cddr, etc.

If you need additional helper functions, *define* them.