CS 3721: Programming Languages Lab
Lab #01: Basic Scheme: learn a new language.

First, start the DrScheme programming environment. The environment has been installed on the Linux machines of the Computer Science department and you can start it by typing the "drscheme" command. You can also download *DrScheme* from *http://www.drscheme.org* to your own machines. The web site *http://www.drscheme.org* has comprehensive documentation both about *DrScheme* and about the *Scheme* language itself.

After the *DrScheme* environment starts, you will see two windows: the top window (called *definition window*) is for collectively entering programs and saving commands into files; the bottom interactive window is for experimenting with Scheme commands without saving them (your commands in the interactive window can use definitions in the definition window). **You should save all your code in the *definition window* into a file so that your work is not lost**.

The Scheme language itself can have a few variations. For the purpose of our class, we would like to choose the language level to be "Teaching Languages: Intermediate Student with lambda" (you can choose the language level when you first start "drscheme" or by clicking the menu item "language:choose language").

The purpose of this class is to lead you through the process of building simple Scheme programs. In a Scheme program, any line that begins with one or more semicolons is a comment:

```
; This is a comment.  DrScheme will store it along with the rest of the
; text of a program, but does not even attempt to execute any part of it.
```

You should finish the following exercises by first experimenting in the interactive window, then saving your solution in the definition window with a comment above to indicate the corresponding exercise. If the exercise asks a question that needs to be answered in English (not in Scheme), provide your solution in the Scheme comments as well. **After saving the solution for each question, click the 'Run' button at the top of the definition window to run your code.** There should be no error messages.

In the following, we start with the most primitive concepts in Scheme.

1. What types of atomic values does Scheme support? (symbol, number, and boolean) At the interactive window, try out examples of constant symbols and numbers. Write down two examples for each type.
2. What type of compound data structures does Scheme support (list)? At the interactive window, try to build lists of symbols and numbers, as well as nested lists. Give an example of an empty list, a single non-empty list, and a nested list respectively.
3. The syntax of Scheme operations. Every Scheme program contains a sequence of expressions, each expression could be a single constant value, a variable, or an operation followed by a number of other expressions (operands). The syntax of each operation has the following format: '(' op exp1 exp2 ... ')' — that is, each operation starts with '(', followed by an operator (e.g., +,-,*,/), then followed by one or more operands, and finally ends with ')'. Give a Scheme expression that evaluates $(59 + 8.2) * 2.3/1.5$. What is the result of the expression?
4. You can similarly apply comparison operations such as <, >, <=, >= to numbers. Give an example for each of the above operators.

5. You can apply *and*, *or*, and *not* operators to boolean values (which are either $true$ or $false$). Give an example for each of the above operators.
6. In Scheme, to declare a new global variable, use operator $define$, which takes two operands, the name of the variable, and the value of the variable. In the interactive window, define a variable x which has number 123.45 as value. Write down your definition. What happens if you try to redefine the value of x?
7. Scheme provides three operators (cons, car, and cdr) for dynamically building lists and for pulling lists apart. Try to build a list which contains three elements: 5, 'abc, and 17. Save the resulting list in a variable named $myVar$. Then try to pull apart the list $myVar$ using *car* and *cdr*. Write down your expressions.
8. What operators can be used to dynamically determine the types of variables in Scheme? (symbol?, number?, boolean? null? and cons?). Give an example for each operator.
9. You can apply the *eq*? operator to determine whether any pair of operands are identical. Give an example using the *eq*? operator. What is the result of *(eq? '(1 2 3) '(1 2 3))*?
10. In Scheme, function can be defined as a variable that has a *lambda* value. For example, *(define id (lambda (x) x))* defines a function which takes a single parameter x and returns x as result. Define a function named *add* which takes two parameters, x and y, and returns the result of $x + y$ (using the + operator).
    You can then invoke the function by typing *(add 3 1)* in the interactive/definition window. Give a few examples of invoking your function.

Submission instructions: To submit, go to

`http://www.cs.utsa.edu/~cs3723`

and fill in the proper information. The instructor will give you a one-time username and password to submit successfully. Please make sure you fill in your name and email address correctly. Such information will be used to create a more permanent submission account for you.