Language Assignment #1: Scheme

Issued: Tuesday, September 6
Due: Tuesday, September 27

Purpose

This assignment asks you to begin using a functional programming language named Scheme, which is a modern dialect of the venerable language Lisp. Lisp was designed by John McCarthy, at MIT, in 1958. Scheme was designed by Guy Steele and Gerald Sussman, at MIT, in 1975.

Submission

Homework is due at 11:59PM, Mountain Time, on the day it is due. Late work is not accepted. To submit your solution to an assignment, login to a lab computer, change to the directory containing the files you want to submit, and execute:

```
1  submit buff class assignment
```

For example:

```
1  submit buff cs101 hw1
```

The `submit` program has a nice `man` page.

Documentation

Scheme lecture slides are at:

```
1  ~buff/classes/354/pub/slides/slides-scheme.pdf
```

Scheme is described in Section 11.3 of our textbook.

The onyx cluster has a Scheme interpreter, the documentation of which can be viewed by:

```
info mit-scheme-ref
info mit-scheme-user
```

and demonstrated by:

```
~buff/classes/354/pub/sum/scheme
```

This documentation, in HTML, is also at:

```
http://www.gnu.org/software/mit-scheme/documentation/mit-scheme-ref
http://www.gnu.org/software/mit-scheme/documentation/mit-scheme-user
```

Assignment

Write and fully demonstrate a function named `replace`, with this interface:

```
(replace source search-for replace-with)
```

The function returns a *copy* of `source`, with every instance of an object that matches `search-for` replaced by a *copy* of `replace-with`. Each argument can be an atom or a list. If no matches are found, the function simply returns a *copy* of `source`. If you are unsure of whether you are making a copy, you might look at:

```
~buff/classes/354/pub/la1/test-if-copy.scm
```

For example:

```
(replace 1 1 2)
⇒ 2
(replace '(a (b c) d)
         '(b c)
         '(x y))
⇒ (a (x y) d)
(replace '(a (b c) (d (b c)))
         '(b c)
         '(x y))
⇒ (a (x y) (d (x y)))
(replace '(a b c)
         '(a b)
         '(x y))
⇒ (a b c)
(replace '(a b c)
         '(b c)
         '(x y))
⇒ (a x y)
```

Of course, you can define other functions and call them from `replace`. For example, you are expected to define and call a function named something like `copy`, which returns a newly allocated copy of its argument.

You are required to use only the *pure* subset of Scheme:

- no side-effecting functions, with an exclamation mark in their names (e.g., `set-car!` and `set-cdr!`)
- no loops (e.g., `do`, `foreach`, and `map`)

Test your solution thoroughly. The quality of your test suite will influence your grade.

Finally, do not try to find a solution on the Internet. You'll possibly be asked to solve a similar problem on an exam, and if you have not developed a solution on your own, you will not be able to do so on the exam.