CISC280 Spring 2007 Lab 8

The Scheme function **read** takes no arguments, and takes user input from the keyboard. There is a note on it on page 383 of your text. Play with it in the interpreter until you are comfortable predicting its behavior with a variety of inputs.

Then try using it inside simple procedures for user interaction. For example, explain the behavior of this procedure:

```
(define (bad-way-to-read)
  (cond ((eq? 1 (read)) 'tuna)
        ((eq? 2 (read)) 'spam)
        (else 'croquet)))
```

Once you are comfortable with read, consider a user interface that allows the user to control the order of icons on a toolbar. Each icon is associated with a function that gets called when the icon is clicked. Our job is to re-order the icons. Write the function change-proc-menu-order so that it behaves as follows:

```
(define assoc-list (map cons '(square fib fact) (list square fib fact)))
> assoc-list
((square . #<procedure:square>) (fib . #<procedure:fib>) (fact . #<procedure:fact>)
> (change-proc-menu-order assoc-list)
Current order: (square fib fact)
Type a new list order to rearrange menu: (fib fact square)
Type 1 to change order, 2 to return list: 1
Current order: (fib fact square)
Type a new list order to rearrange menu: (fact fib square)
Type 1 to change order, 2 to return list: 2
((fact . #<procedure:fact>) (fib . #<procedure:fib>) (square . #<procedure:square>)
>
```

If you need to submit (see syllabus if you are unsure) submit your code file(s) and a script of several well-chosen test cases via MyCourses (due Thursday midnight) and on paper (to your TA's mailbox Friday by 1 p.m.) to receive full credit.

When you use MyCourses, remember that you can "upload" files multiple times, but you only click "submit" once.