CISC280 Spring 2007 Lab 3

Read the file quote.txt in this directory. You will not need it to write this assignment, but you will need to know its contents tomorrow, and reading it now gives you an opportunity to play and ask questions about it in lab.

In class we wrote two procedures, **scale** and **add**. Then we looked at them and saw they were very similar in structure. We were able to abstract away from a specific operation and write map, so that we could reproduce `(scale 5 (list 1 2 3))` by saying

```
> (map (lambda (x) (* x 5)) (list 1 2 3))
```

and without needing a special scale function. Indeed, **map** works well any time we wish to do the same thing to each member of a list.

Write the two functions **sum** and **product** so that they behave as below. What must the base case be for sum to work correctly? for product?

```
> (sum '(1 2 3 4))
10
> (product '(1 2 3 4))
24
```

Now, as we did with map, write a linear recursive function **reduce** that can perform both sum and product operations on a list. You will have to add one or two parameters so that you can specify the behavior. One parameter will be the operator required, either + or $*$.

After you have written reduce, consider how the operator argument is applied each time through the procedure; in particular, what is on the left, and what is on the right? See if you can write a lambda procedure to pass as an argument to reduce that will leave the list argument unchanged.

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.