# Lab 2

Submit either .scm files (for code) or text files (for written answers to questions) for each of the following. **All** code files for this lab must contain proper tests that run when the file is loaded.

1. Code iterative and recursive process versions of length and nth.
2. Do A&S 1.12 (Consider each row a list; work on getting from the nth row to the (n+1)st row).
3. Code the procedures `append` and `reverse`. Code each one twice: process recursively and iteratively. Note: an iterative procedure that calls a recursive process helper is no longer iterative; but it may certainly call other iterative procedures.
4. Code and time expt, fast-expt. Use the time-apply[1] function in mzscheme to time the two functions on the same data, and graph your results using software, e.g. Matlab. Use multiple runs on the same data points to minimize the effects of outliers; graph all points.
5. Xenon wants to write a decimal-to-binary converter in Scheme. S/he knows about what columns in number systems mean about values, but cannot figure out how to use that information to convert a number in a program without writing something really inefficient.

   Then Zorpox shows Xenon an algorithm for computing binary numbers that does not involve lots of exponentiation and comparison, but only uses division by two. It works like this:

   (a) If n is less than 2; write n.

   (b) Divide n by 2; if there is a remainder, write 1. If there is no remainder, write a 0.

   (c) Recurse on what is left of n after the division.

   Xenon happily[2] implements this algorithm, and gets the following results:

```
> (convert 0)
0
> (convert 4)
001
> (convert 13)
1011
```

[1] Information about this function is available online.

[2] Well s/he wasn't happy at first; then s/he remembered two things: 1) you can get Scheme to "write" something with the **display** command, and 2) when you are using a **cond**, you can have more than one expression be evaluated for a single true condition.

**Write** Xenon's recursive convert program using **only** the following scheme operators: lambda, =, <, quotient, remainder, display, newline.

Xenon can tell s/he is close to the solution, but that the output is reversed. Xenon wants to fix the problem, but knows that the printed results of **display** cannot be turned around. Then Xenon realizes that the placement of the calls in the program can be changed, and that will change the output. After a little fiddling Xenon gets:

```
> (decimal2binary 0)
0
> (decimal2binary 1)
1
> (decimal2binary 127)
1111111
> (decimal2binary 128)
10000000
>
```

**Modify** your program to produce the correct result without adding any new operations (but you may add a wrapper).

Submit your code file and a script file showing files being loaded (and tested) via Sakai (due Sunday midnight) and on paper (to your TA at the START of **Monday** lab) to receive full credit.

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