

CIS-280
Lab 3: Recursive versus Iterative Processes
Monday, Feb. 28, 2005

This lab will explore the difference between recursive and iterative processes.

1. First you want to change your language level to **Beginner** so that you can use the stepper. Do the following:
 - Click on **Language** at the top of the Scheme window and select **Choose language**.
 - On the new window that appears, click on **How to Design Programs** under **Teaching Languages**, and then select **Beginning Student**.
 - Click on **OK** at the bottom of the window. The window will close.
 - Now click on **Execute** at the top of your Scheme window. In the Interpreter window, you should get a message that says **Language: Beginning Student**.
2. Copy file `~carberry/Lab3a.scm` (containing the following procedure for computing exponents) into a file in your directory and then load the contents of the file into scheme.

```
(define expt1 (lambda (number n)
; computes the value of number taken to the power n
  (cond ((= n 0) 1)
        (else (* number (expt1 number (- n 1)))))))
```

When executed, `expt1` is a recursive process (expansion of delayed computations followed by contraction), with execution time that is $O(n)$ where `n` is the power to which `number` is to be computed. Thus it is said to be a linear recursive process.

3. Click **Execute** at the top of the Scheme window to define the procedures.
4. Now we want to view the expansion and contraction that occurs with a recursive process. Do the following:
 - Type `(expt1 3 5)` in the Definition window.
 - Then use the cursor to highlight `(expt1 3 5)`
 - Now click on the **Step** button at the top of the Scheme window.
 - In the new window that appears, click on **Step** repeatedly and observe what happens. Although you will see each step of the execution, what you want to observe is the expansion that occurs with each new call to `expt`. For example, after several steps, you will see `(* 3 (expt1 3 4))` and then several steps later you will see `(* 3 (* 3 (expt1 3 3)))`. Observe the sequence of expansions that occur, and then the contractions as the delayed computations are eventually performed.
5. Now copy file `~carberry/Lab3b.scm` (containing the following procedures) into a file in your directory and then load the contents of the file into scheme.

```
(define expt2 (lambda (number n)
; computes the value of number to the power n
; this procedure has linear time and space
  (expt2-iter number n 1)))
```

```
(define expt2-iter (lambda (number n answer)
  (cond ((= n 0) answer)
        (else (expt2-iter number (- n 1) (* answer number))))))
```

When executed, `expt2` is an iterative process (state variables in procedure `expt2-iter` capture the status of the computation at any point in time), with execution time that is $O(n)$ where `n` is the power to which `number` is to be computed. Thus it is said to be a linear iterative process.

6. Click **Execute** at the top of the Scheme window to define the procedures.
7. Now we want to view the difference between a recursive and an iterative process. Do the following:
 - Type `(expt2 3 5)` in the Definition window.
 - Then use the cursor to highlight `(expt2 3 5)`
 - Now click on the **Step** button at the top of the Scheme window.
 - In the new window that appears, click on **Step** repeatedly and observe what happens. Although you will see each step of the execution, what you want to observe is the sequence of calls to `expt2-iter`. For example, in the beginning you will see `(expt2-iter 3 5 1)`; then after several steps, you will see `(expt2-iter 3 4 3)` and then several steps later you will see `(expt2-iter 3 3 9)`. Observe how the three parameters of `expt2-iter` change and play the role of state variables that capture the current status of the computation at any point during execution — that is, all the information needed to resume the process is given in these variables. Contrast this with the first process, where some of the information is hidden in the procedure (namely, the delayed computations that must be executed once each called procedure returns a result).
8. Design and implement two procedures `NumberDigits1` and `NumberDigits2` that take a positive integer as argument and return the number of digits in the argument. For example, if the argument is 45832, then both procedures return 5 since the argument has five digits. Both `NumberDigits1` and `NumberDigits2` should be recursive procedures, but `NumberDigits1` should be a linear recursive process and `NumberDigits2` should be a linear iterative process.
9. Copy your code for `NumberDigits1` and for `NumberDigits2` into the electronic submission system, and submit it for Lab3.
10. Before you leave lab, change your language level back to **Graphical (MrEd, includes MzScheme)** by doing the following:
 - Close all but the first Scheme window that was opened.
 - Click on **Language** at the top of your initial Scheme window, then select **Choose language**.
 - In the small window that appears, click on **PLT** under **Professional Languages** and then select **Graphical (MrEd, includes MzScheme)**.
 - Click on **Execute** at the top of your Scheme window. A message should appear in the Interpreter window saying `Language:Graphical (MrEd, includes MzScheme)`.