Lab #2: Functions and Recursion

CSC 358/458, Prof. Robin Burke
Spring 2006
In-class: 4/10/2006 -- Distance learning: 4/17/2006

Objective:
Use functions, closures and recursion.

What to do:
1. Create a file lab2.lsp and in it, write a defstruct declaration to create a structure for a Java-style iterator with has-next and next as slots.
2. Write a function truth that returns an iterator structure that always returns the value T.
3. Write a function integers that returns an iterator structure that returns the non-negative integers.
4. Here is a function that will put the first n results of an iterator in a list.
   (defun list-n (iter n)
     (let ((ans nil))
       (dotimes (i n)
         (if (funcall (iterator-has-next iter))
           (setf ans (append ans
                        (list (funcall (iterator-next iter)))))))))
     ans))
   Write a recursive version of this function called list-n-rec.
5. Write a tail-recursive version of list-n called list-n-tail. Use a subfunction list-n-tail-1.
6. (Extra credit) Write a function filter-it-if that takes an iterator and a predicate and returns a "filtered" version of the iterator one that includes only those items for which the predicate answers T. For example, (filter-it-if #'evenp (integers)) should return an iterator consisting of only the even integers.

What to turn in:
Submit lab2.lsp to the DLWeb website for the course.

Hints and Notes:
- Use funcall to call the functions stored in the iterator structure.
- Use lexical closures with abandon.
- You may need to use prog1 to calculate a return value and then do additional computation: (prog1 form1 ... forms), form1 is evaluated and the result is stored. The other forms are all evaluated and then when the form exits the value of form1 is returned.