Local Procedures and Recursion

- **Exercises**
  - Exercise 0: Preparation
  - Exercise 1: The Last element
  - Exercise 2: Alternating lists
  - Exercise 3: Iota, Revisited
  - Exercise 4: Taking Some Elements
  - Exercise 5: Intersection

- **Notes**

**Exercises**

**Exercise 0: Preparation**

a. Review the corresponding notes on `letrec` and named let.

b. Start DrScheme.

**Exercise 1: The Last element**

Write a `letrec`-expression in which (a) the identifier last-of-list is locally bound to a procedure that finds and returns the last element of a given list, and (b) the body of the expression computes the sum of the last elements of the lists (3 8 2), (7), and (8 5 9 8), invoking last-of-list three times.

**Exercise 2: Alternating lists**

A non-empty list is an *s-n-alternator* if its elements are alternately symbols and numbers, beginning with a symbol. It is an *n-s-alternator* if its elements are alternately numbers and symbols, beginning with a number.

Write a `letrec`-expression in which (a) the identifiers *s-n-alternator?* and *n-s-alternator?* are bound to mutually recursive predicates, each of which determines whether a given non-empty list has the indicated characteristic, and (b) the body invokes each of these predicates to determine whether the list (2 a 3 b 4 c 5) fits either description.

**Exercise 3: Iota, Revisited**

As you may recall, *Iota* takes a natural number as argument and returns a list of all the lesser natural numbers in ascending order.
a. Define and test a version of the \texttt{iota} procedure that uses \texttt{letrec} to pack an appropriate kernel inside a husk that performs precondition testing.

b. Define and test a version of the \texttt{iota} procedure that uses a named \texttt{let}.

\textbf{Exercise 4: Taking Some Elements}

Define and test a procedure named \texttt{take} that takes a list \texttt{ls} and a non-negative integer \texttt{len} as arguments and returns a list consisting of the first \texttt{len} elements of \texttt{ls}, in their original order. The procedure should signal an error if \texttt{ls} is not a list, if \texttt{len} is not an exact integer, if \texttt{len} is negative, or if \texttt{len} is greater than the length of \texttt{ls}.

\textbf{Exercise 5: Intersection}

a. Define and test a procedure named \texttt{intersection} that takes two lists of symbols, \texttt{left} and \texttt{right}, as arguments and returns a list of which the elements are precisely those symbols that are elements of both \texttt{left} and \texttt{right}.

b. What does your procedure do if a symbol appears in both lists and appears more than once in one or both of the lists?

\textbf{Notes}