EECS 345 Homework #5 - due 11/8/05

1. Implement COND in the Tiny-2 interpreter as a syntactic transformation to a nested IF expression, which is then passed to TEVAL to be evaluated. Note that, in Tiny, the default case should use $TRUE rather than Common Lisp's T.

2. Closures can be used to implement data structures. Use this approach to add lists to Tiny-2. **Note that you are to implement lists IN Tiny, not to add lists to Tiny's underlying Common Lisp implementation**. A minimal set of operations for lists are `CONS`, `CAR` and `CDR`. These operations should respect the obvious identities `(CAR (CONS X Y))` ⇒ `X` and `(CDR (CONS X Y))` ⇒ `Y`. You will also need to define a global constant `$NIL` to represent the empty list. `(CAR $NIL)` and `(CDR $NIL)` should both return the empty list. Once your implementation is complete, you should be able to write and run simple recursive list-processing functions, such as `LENGTH` or `REVERSE`, in Tiny:

   ```
   (define (length list)
     (if (eq list $NIL)
         0
         (+ 1 (length (cdr list)))))

   (length (cons 10 (cons 9 (cons 8 $NIL)))) ⇒ 3
   ```

   Hint: `CONS` should return a closure with closed variables holding the car and the cdr for that cell. `CAR` and `CDR` should access those closed variables. `$NIL` will need to be bound to a closure whose `CAR` and `CDR` are itself.

3. Add AND with Boolean short-circuiting to Tiny-2, so that, for example, `(if (and (> x -1) (< x 1)) 1 0)` would work, and the second comparison would only be evaluated if the first comparison was true.