Fall 2003 **LISP Written Assignment #2** CMSC 420

Hanan Samet

The following exercises are designed to test your understanding of recursion. The functions are defined using a variant of LISP known as meta-LISP. In order to aid your understanding, the function defined in problem 1 is identical to the one below:

```
drop(x) =  if null x then nil
           else (car x) cons drop(cdr x)
```

The idea is that
a x = car x
d x = cdr x
n x = null x
at x = atom x
a.b = a cons b
<a> = a cons nil = a list whose single element is a
a*b = concatenate lists a and b (i.e. append list b to list a)
reverse[x] = reverses the top level list x. For example reverse[(A B C)] = (C B A). But reverse[((A B C)(D E))] = ((D E)(A B C)).

1. Consider the function drop defined by

   ```
   drop[x] ← if n x then nil else [a x].drop[d x].
   ```

   Compute (by hand) drop[(A B C)]. What does drop do to lists in general?

2. What does the function

   ```
   r2[x] ← if n x then nil else reverse[a x].r2[d x]
   ```

   do to lists of lists? How about

   ```
   r3[x] ← if at x then x else reverse[r4[x]]
   r4[x] ← if n x then nil else r3[a x].r4[d x]?
   ```

3. Compare the following function with the function r3 of the preceding example:

   ```
   r3'[x] ← if at x then x else r3'[d x]*<r3'[a x]>
   ```

4. Consider r5 defined by

   ```
   r5[x] ← if n x ∨ n d x then x
           else [a r5[d x]] .  r5[a x .  r5[d r5[d x]]].
   ```

   Compute r5[(A B C D)]. What does r5 do in general. Needless to say, this is not a good way of computing this function even though it involves no auxiliary functions.