COSC 341 Programming Project #2 LISP

Distributed: 17 February 2009 Due: 5 March 2009

Write three lisp programs as specified below. These programs can all appear in the same file.

1. You need to explode some atoms. Unfortunately, Common Lisp does not include functions for exploding atoms. I've found souce code, though, `at http://www.cis.udel.edu/~mccoy/courses/cisc4-681.08f/programs/graphsearch.lisp` that you can use (properly cited, of course, to explode atoms.

```
(defun implode (list)
    ;;; takes a list of single character atoms and combines them into a
    ;;; single atom. Common lisp has no built-in implode function.
    (cond ((null list) nil)
        ((read-from-string
          (concatenate 'string
                   (mapcar #'(lambda (x)
                             (cond ((numberp x)(digit-char x))
                                 ((character x))))
                         list))))))

(defun explode (atom)
   ;;; takes an atom and splits it into a list of single character
   ;;; atoms. Common lisp has no built-in explode function.
    (cond ((listp atom) atom)
        (t
         (let ((s (string atom)))
           (explodes s 0 (length s))))))

(defun explodes (str position len)
   ;;; recursive helping function for explode
   (cond ((eq position len) nil)
       ((cons (read-from-string (string (schar str position)))
            (explodes str (1+ position) len)))))
```

Write a Lisp function called `check-particles` that takes two arguments, both atoms. Return a list of all characters that appear in both atoms. Nil should not appear as a member of that resulting list. Use and write as many "helper" functions as you reasonably can.

Extra credit for no redundant characters and no nils in the result.

2. Write a recursive function named `nn` that returns 1 where each element of a list is non-nil and 0 where the elements are nil. For example, `(nn '(1 () (2 3) (nil nil) 1))` returns `(1 0 1 1 1)`

3. Write a recursive function named `count-atoms` that counts the number of atoms that appear at all levels of a list. For example, `(count-atoms ‘(a b (c (d (e) f) (g) ) h))` returns `8`.

4. Extra credit. Need more Lisp? Implement mergesort in Lisp. This function takes one argument that is a list of integers and returns a sorted list of those integers.

The pseudo-code for this sort routine is available in any data structures book. Here it is:

```
mergesort (array) {
     if (length of array is 1) return array;
     else {
          mergesort (left half of array);
          mergesort (right half of array);
          merge (sorted left half , sorted right half);
          return merged array;
          }
     }
```

The merge function takes two sorted arrays and produces one array of sorted values.

Turn in:

- You’ll need to demo these programs. We’ll organize demos later.
- Hard list of programs
- Hard copy screen shot of simple demo of each program.
- Email versions accepted ONLY on the due date itself.

Grading:

Meets specifications	80%
Meets style standards	10%
Code elegance	10%

Late projects accepted up to 10 March 2009. Late hit of 10% (flat rate) from 5:01pm 3/5/09 until 5:00pm 3/10/09.