Name: _____

4 questions in 120 minutes

Question 1 (20%)

Define a recursive function that counts the non-nil atoms in a list. For instance, an input like `((a b) c)` should return 3, `(a ((b (c) d)))` should return 4, and so on. Remember that the built-in `ATOM` returns `NIL` for all lists except `NIL`; `NULL` returns `T` only for `NIL`; `ENDP` is like `NULL`, except that it gives an error if its input happens to be something other than a list. Your function should use a counter/accumulator – it will be a two argument function.

Name: _____

Question 2 (30%)

Define a function `BRING-TO-FRONT` (or `BFT` for short), that takes an item and a list and returns a version where all the occurrences of the item in the given list are brought to the front of the list. For instance, `(bring-to-front 'a '(a b r a c a d a b r a))` would return `(A A A A A B R C D B R)`; and `(bring-to-front 'b '(a b r a c a d a b r a))` would return `(B B A R A C A D A R A)`. You are NOT allowed to count the occurrences of the item in the given list or use `REMOVE`.

Name: _____

Question 3 (30%)

Define a function that groups the elements in a list putting consecutive occurrences of items in lists. For instance, `(group '(a a b c c c d d e))` should give `((A A) (B) (C C C) (D D) (E))`. Note that you should NOT bring together non-consecutive repetitions; a call like `(group '(a b b c b b c))` should return `((A) (B B) (C) (B B) (C))`.

Name: _____

Question 4 (20%)

Here is a recursive function that removes the last occurrence of an element from a list:

```
(defun rem-last (x lst &optional backup guess)
  "Remove the last occurrence of x in lst"
  (cond ((endp lst) guess)
        ((equal x (car lst)) (rem-last x (cdr lst)
                                (append backup (list x))
                                backup))
        (t (rem-last x (cdr lst)
             (append backup (list (car lst)))
             (append guess  (list (car lst)))))))
```

Here is a similar function that removes the one before the last occurrence:

```
(defun rem-pen (x lst &optional backup guess1 guess2)
  "remove the one before the last occurrence of x from lst"
  (cond ((endp lst)
         (if guess2 guess2 backup))
        ((equal x (car lst))
         (rem-pen x (cdr lst)
                  (append backup ?1)
                  ?2
                  (append ?3 (list x))))
        (t
          (rem-pen x (cdr lst)
                   (append backup (list (car lst)))
                   (append guess1 ?4))
                   (append guess2 (list (car lst)))))))
```

Specify the LISP expressions that should come in place of ?1, ?2, ?3 and ?4.