Spring 1997 Due: 3/18

Programming Project #3: Writing Simple lisp functions

CMSC 420 David Traum

Write lisp functions to implement each of the following.

1. `(alt2 list)`, which returns every other item in a list, skipping the first one (if any).
2. `(count item list)` which returns the number of times `item` is seen in `list` (hint: use a help function with a variable to count the number of times you've seen `item` so far)
3. `(remove_last list)` which returns a list including all but the last item of the input list.
4. `(len list)` which returns the number of items in (the length of) a list.
5. `(nth_item n list)` which returns the nth item of the list. The car is the 1st item. Return the special symbol `*error*` if there are less than n items in `list`.
6. `(remove_nth n list)` which returns a list with the nth item removed from the original list. Return the special symbol `*error*` if there are less than n items in `list`.
7. `(replace_nth new n list)` which returns a new list with `new` in the place of the old nth item. Return the special symbol `*error*` if there are less than n items in `list`.
8. `(pos item list)` which returns the position of the item in the list. Return nil if the item is not in the list. The first item in the list is in position 1.
9. `(sum_nums list)` return the sum of all the numbers in the list.
10. (a) `(average list)` return the arithmetic mean of the numbers. This is the sum of the numbers divided by the number of items in the list.

    (b) if you did part (a) the easy way, calling the functions `sum_nums` and `len`, you are making two recursive calls on the list. Can you rewrite this function (using helper variables) so that you only have to process each item in the list once?
11. It is often useful to perform operations on compound data structures, such as a list of lists. We will call this data structure `lol`. For this and the next three problems, you can assume that all lists have the same number of items.

    `(allcars lol)` return a list of all the first items of the sublists
12. `(all_nths lol)` return a list of all the nth items of the sublists. Use the function `nth_item` that you wrote in (5).
13. `(find_lol item lol)` find an item in any of the lists
14. One use for a list of lists is to represent a matrix, where the lists are the rows, and the items in the list the respective items in the row. This is row major order. Write a function to transpose the matrix into column major order, where you return a list of lists, where now the first list is the first row, the second list the second row, etc. You may use any of the functions you implemented above to simplify this function.