Assignment Three – Beginning Lisp Programming

November 17, 2014 — Due December 1, 2014

Objective

To understand syntax, semantics, and programming styles of language Lisp. Familiar to recursive thinking. Practice functional programming paradigm.

Problem Specification

1. Write a function `dispnth` to display the n-th element of a list. You may assume that the input list always has n or more elements.
   `(dispnth '(1 (2 3) 4 5) 2)` → `(2 3)`
2. Write a function `delnth` to delete the n-th element of a list. You may assume that the input list is always longer than n.
   `(delnth '(1 2 (3 4) 5) 3)` → `(1 2 5)`
3. Write a function `lastele` to display the last element of a list, or `NIL` if the list is empty.
   `(lastele '(1 (2 3) 4 5))` → `5`
4. * Write a function `lastele2` to display the last list of a list, or `NIL` if the list is empty or no list element exists.
   `(lastele2 '(1 (2 3) 4 5))` → `(2 3)`
5. Write a function `remv` to remove given single elements from a list (including multiple appearance).
   `(remv a '(a (b) a c))` → `((B) C)`
6. * Write a function `remv2` to remove given list elements from a list (including multiple appearance).
   `(remv2 '(a b) '(a b (a b) c))` → `(A B C)`
7. Write a function `remvdub` to remove duplicate elements from a list.
   `(remvdub '(a b a c b a))` → `(A B C)`
8. * Write a function `remvdub2` to remove duplicate elements (single element or lists) from a list.
   `(remvdub2 '(a b (a) c b (a)))` → `(a b c (a))`
9. * Write a function `lists` to return the list elements of a given list.
   `(lists '(1 (2 3) (4) 5))` → `((2 3) (4))`
10. Write a function `inde` which returns the index (start from 1)of the occurrence of a given value.
    `(inde 1 '(1 2 1 1 2 2 1))` → `(1 3 4 7)`
11. Write a function `nele` which repeats each element in a list n times.
    `(nele '(1 2) 3)` → `(1 1 1 2 2 2)`

12. Write a function `istrin` to determine if a positive integer is a triangular number.
    `(istrin 21)` → `T`

13. Write a function `model` which returns the mode and its occurrence of a given list. Higher order function `filter` (you have to provide one) is required.
    `(model '(1 3 5 2 3 5))` → `((3 2) (5 2))`

14. Write a function `permu` to generate the permutation of the identity list from 1 to n. Higher order function `mapcar` is required.
    `permu(3)` → `((1 2 3) (2 1 3) (2 3 1) (1 3 2) (3 1 2) (3 2 1))`

What To Do

- Your functions must use the precise given names.
- Do not use any Lisp build-in library functions.
- You must use recursion to write functions.
- Helper functions are allowed. However, try **NOT** to use `append` and **NOT** construct result in function argument.
- Your program should be well documented and written in a nice layout.
- Each function should be documented with a function header as follows:

```
;;; FUNCTION NAME:   <name of the function>
;;; DESCRIPTION:    <description of the function>
;;; NOTES:          <comments of the function>
```

- Prior to class of December 1, submit your zip electronically.
- All your functions should be put into one file named by `your first name, last name`, and `03.lisp`. For example, my submission is looked like `lizhang03.lisp`. Then zip it when you submit.
- Your file should contain plain code only. **No comments** or test runs.
- The hard copy of a written formal report which contains at least
  - Nicely layout format with a cover page including your name and E-number.
  - Any comments regarding the assignment which you want to address.
  - Source code with comments as appendix.
  - Sample runs **as they are**.

  The report will be collected at the beginning of class of December 1. No late submission.