CS324 SPRING 2014 **Lisp** Programming Assignment

(Total 150 pts)

Due:before the end of *Friday, May 2nd, 2014*

Write the following functions in *CMU Common Lisp 20a Fedora release 1.fc13 (20A Unicode)*, running on the NMT TCC "rainbow.nmt.edu" machine:

1- insert(X L): a function that takes a list L and an item ***x*** and returns L after the insertion of the item ***x*** at the end of L. *(15 pts)*

Example: **(insert '4 '(1 2 3))** should return **(1 2 3 4)**.

2- insert-after(*x y* L): a function that takes an item ***x*** (to be inserted) and an item ***y*** (used as a position), and a list L; returns L with ***x*** inserted after the first occurrence of the item ***y***. If ***y*** is not in L, then it returns L. *(15 pts)*

Example: **(insert-after '4 '5 '(1 2 3 5 7 9 8 5 3 6))**
should answer --> **(1 2 3 5 4 7 9 8 5 3 6)**
(insert-after '4 '5 '(1 2 3)) --> returns (1 2 3)

3- my-remove(X L): a function that takes a list L and an item ***x*** and returns L after the removal of the first occurrence of item ***x*,** otherwise if x is not in L, then it returns L. *(15 pts)*

Example: **(my-remove '4 '(1 2 3 4 5 8 4))** should return **(1 2 3 5 8 4)**.

4- plus-map(L1 L2): a function that adds the corresponding elements of two lists L1 and L2 and returns a list of the results. *(15 pts)*

Example: **(plus-map '(1 2 3 4) '(1 9 8 4))** should return **(2 11 11 8).**

5- my-count(L): a function that counts the number of **atoms** an input list L no matter of their nesting levels. *(30 pts)*

Example: **(my-count '(a b (c 4) ((99)) nil t))** should return **7**.

6- filter(P L): a function that takes as input: a predicate **P** and a list L and returns a list of only those elements of **L** that satisfy **P**. *(15 pts)*

Example: **(filter 'minusp '(2 –3 7 –1 –6 4 8))**
should return **(-3 –1 -6).**
Also, **(filter 'listp '(a (b c) d (e f g)))** should return **((b c) (e f g)).**

7- my-reverse(L): a function that returns it input list L in a reverse order. *(25 pts)*

Example: **(my-reverse '(1 2 3 4 5))** should return **(5 4 3 2 1)**

8- add-lists-intgrs(L1 L2): a function that output the result of adding the elements of two input lists L1&L2 (of integers), in case of either list is **longer** than the other, the remaining difference of integers will be just concatenated at the end of the output list. *(20 pts)*

Examples: **(add-lists-intgrs '(1 2 3 4 5) '(5 6 7 8 9))**
should return as an output: **(6 8 10 12 14)**
(add-lists-intgrs '(1 2 3 4 5) '(5 6 7 8 9 11 12 13))
should return as an output: **(6 8 10 12 14 11 12 13)**
(add-lists-intgrs '(1 2 3 4 5 11 12 13) '(5 6 7 8 9))
should return as an output: **(6 8 10 12 14 11 12 13)**

Extra Credit: (*35 pts each*)

9) Given "the following definition of an employee "*Attribute*" list (EAL):

((firstname Don) (lastname Smith) (country USA) (gender M) (age 45)
(Salary 30000) (Hire-Date (August 25 1980)))

Define a recursive function "delattribute (*EAL atr*)" which searches the attribute *atr* in the attribute list *EAL* and delete it form and returns the new *EAL* without the deleted attribute.

Example: **(on the EAL defined above)**

%delattribute (*EAL 'gender*)

%((firstname Don) (lastname Smith) (country USA) (age 45) (Salary 30000)
(Hire-Date (August 25 1980)))

%delattribute (*EAL ' nothere*) ;; if the *atr* is not there just return the EAL

%((firstname Don) (lastname Smith) (country USA) (gender M) (age 45)
(Salary 30000) (Hire-Date (August 25 1980)))

10) Assuming the same definition of the employee *Attribute* list (*EAL*) as in problem 9 above, define a recursive function:

"chg-atr-val (*EAL atr newval*)"

which searches the attribute *atr* in the attribute list *EAL* and changes its value to the new value: *newval*.

Examples: **(on the EAL defined above)**

% chg-atr-val (*EAL 'age '55*)

((firstname Don) (lastname Smith) (country USA) (gender M) (age *55*)
(Salary 30000) (Hire-Date (August 25 1980)))

% chg-atr-val (*EAL 'salary '90000*)

((firstname Don) (lastname Smith) (country USA) (gender M) (age 45)
(Salary *90000*) (Hire-Date (August 25 1980)))

% chg-atr-val (*EAL ' nothere 'any*) ;; if the *atr* is not there just return the EAL

((firstname Don) (lastname Smith) (country USA) (gender M) (age 45)
(Salary 30000) (Hire-Date (August 25 1980)))

Warning!!!!!

The usage of *built-in* functions such as **reverse** (to implement reverse-list()), also **remove** (to implement remove-from), and so forth, is **NOT** accepted. All functions should be implemented using **basic built-in** functions such as **car**, **cdr**, **cons**, **append**, …,**last, cond, etc.**

How to run lisp *interpreter*? At the prompt, type: **lisp** then hit return

How to *compile* and use lisp? To compile lisp file "example.lisp", containing some functions (the file extension MUST be **.lisp**) from within the lisp interpreter, write:

* (**compile-file** "example.lisp") which produces a new file: example.x86f

Then to load and execute any function in "example.lisp", at the “lisp” interpreter level:

* (**load** "example.x86f")

From now on, at the system level, you can use any function that you defined and successfully compiled in “example.lisp”. This saves you time of worrying about the matching parenthesis (and other silly mistakes) at the hard to correct system level.

Hints:

- In case of missing up at the command level (e.g., mistyping a command), you will be sent to the debugger. Type **abort** to exit the debugger to the top lisp level.

- To exit the lisp interpreter write **(quit)**.

- Try to minimize writing code at the command interpreter prompt as much as you can, instead write most definitions in the lisp file (<file-name>.lisp) then compile/load it. Anything you defined inside the file will be accessible.

To print an error msg, you may follow this example:

```
(cond ((null x) 'msg: error: undefined-symbole)
( t ....   rest of the code ....        )
```

Remember any text between a ";" and the end of line is considered comments.

Project Submission:

Your project must be submitted on the due date to Moodle as a **plain textfile**. The project's filename must read: "lastname_assignment4.txt"

Warning: If you fail to submit according to the above criteria, your project may not be accepted. If any of the above requirements are not understood, contact the TA for this course.

Note: All work must be your own and failure to do so will result in a zero for the report. All rules in the Academic Honesty Policy strictly apply.