York University- Department of Computer Science and Engineering

SC/CSE 3401 3.00 – Functional and Logic Programming

Solutions to assignment 4

1) (15 marks) write a function with two arguments n and lst, that adds a number n to the elements of lst. You can assume that n is a number and lst is a list of numbers. No checking is required.
For example if lst is '(10 20 30) and n is 5, it will return '(15 25 35).

Write this function in 3 versions:

(a) add2list1: use iteration

```
(defun add2list1 (n lst)
   (do  ((tlst lst (cdr tlst))
         (result nil (append result (list (+ n (car tlst))))) )
      ((null tlst) result)))
```

(b) add2list2: use recursion

```
(defun add2list2 (n lst)
   (cond
      ((null lst) nil)
       (t (cons (+ n (car lst)) (add2list2 n (cdr lst))))))
```

(c) add2list3: use mapcar

```
(defun add2list3 (n lst)
   (mapcar (lambda (x) (+ n x))  lst))
```

2) (35 marks) For the following questions, assume that we write matrices as lists of rows, for example a matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ is represented as '((1 2 3) (4 5 6)).

(a) (5 marks) Write a function that finds the size of a matrix as a list of the form (no_of_rows no_of_cols), for example given a matrix A as '((1 2 3) (4 5 6)), it returns (2 3).

```
(defun getlength (lst)
    (do ((tlst lst (cdr tlst)) (n 0 (1+ n))) ((atom tlst) n)))

(defun getsize(m)
      (list (getlength m) (getlength (car m)) ) )
```

(b) (10 marks) Write a function addm that adds two matrices using mapcar. Check if the matrices are the same size.
Examples:

```
> (addm '((1 2) (3 4)) '((1 0) (0 1)))
 ((2 2) (3 5))
 > (addm '((1 2) (3 4)) '((1 0 1) (0 1 1)))
 Error: Matrices must be the same size!
 NIL

(defun addm (m1 m2)
   (cond
     ((not (equal (getsize m1) (getsize m2)))
           (princ "Error: Matrices must be the same size!") (terpri))
      (t (mapcar (lambda (row1 row2) (mapcar '+ row1 row2)) m1 m2))))
```

(c) (10 marks) Write a function that finds the transpose of a matrix using mapcar. We want this function to work for matrices of any size. For example, given matrix A as '((1 2 3) (4 5 6)) it would return '((1 4) (2 5) (3 6)).

```
(defun transpose (matrix)
   (apply 'mapcar (lambda (&rest l) (apply 'list l)) matrix))
```

(d) (10 marks) Using the transpose function, write a function multm that multiplies two matrices. Check for the correct sizes.

Examples:

```
 > (multm '((1 2) (3 4)) '((1 0 1) (0 1 1)))
 ((1 2 3) (3 4 7))
 > (multm '((1 2) (3 4)) '((0 1) (1 1)))
 ((2 3) (4 7))
 > (multm '((1 2) (3 4)) '((1 0) (0 1)))
 ((1 2) (3 4))
 > (multm '((1 0 1) (0 1 1)) '((1 2) (3 4)))
 Error! Matrices cannot be multiplied, wrong sizes!
 NIL


(defun multm (m1 m2)
   (cond ((not (equal (cadr (getsize m1))  (car (getsize m2))))
          (princ "Error! Matrices cannot be multiplied, wrong
                sizes!") (terpri) )
          (t
                (mapcar (lambda (row)   ; each row of m1
                (mapcar (lambda (col)   ; each col of m2
                (apply '+ (mapcar '* row col))) ; do the calculations
                  (transpose m2)))      ; transpose provides the columns
```

```
                          m1))))
```

---

3) (35 marks) In this question, you will be writing a function **datastat** that returns simple statistics of data collected over temperatures in a location. This function can accept the data in 3 ways: i) can wait for the user to input data, ii) can get the data in a list as an argument, iii) can read the data from a file specified by the user.

For example:

i) Wait for the user to input data:

```
> (datastat)
Enter all data, finish with 'end >10 11 15 14.2 -12 end
Number of data:   5
Average:           7.64
Minimum:        -12.00
Maximum:         15.00
Range:           27.00
NIL
```

ii) Get the list of data as an argument

```
> (datastat :data '(9  7.5 -5 -2))
Number of data:   4
Average:           2.37
Minimum:         -5.00
Maximum:          9.00
Range:           14.00
NIL
```

iii) Read data from file:

```
> (datastat :file "temp.txt")
Reading data from file ...
Number of data:  15
Average:           6.92
Minimum:        -23.23
Maximum:         34.60
Range:           57.83
NIL
```

In writing your function, define any variable you might need in your code as **auxiliary** parameters. Output the results in the format shown in above examples. The "temp.txt" file used in above example can be downloaded from the course webpage. You can assume input is given in correct format, no checking is necessary.

Answer.

```
(defun datastat (&key data file &aux (sum 0) (min +1000) (max -1000) (n 0)
                  instream)
      (cond
           ((null data)
             (cond  (file (setq instream (open file :direction :input))
                          (format t "Reading data from file ...~%"))
                    (t    (setq instream *standard-input*)
                          (princ "Enter all data, finish with 'end >")))
             (loop
                (setq data (cons (read instream nil 'end) data))
                (if (equal (car data) 'end) (return t)))
             (setq data (cdr data))
             (if file (close instream))))

      ; now we have a list of data, calculate
      (dolist (i data t)
            (setq sum (+ i sum))
            (setq n (1+ n))
            (if (< i min) (setq min i))
            (if (> i max) (setq max i)))
      ; report
      (format t "~15a ~3d~%~15a ~6,2f~%~15a ~6,2f~%~15a ~6,2f~%~15a ~6,2f"
            "Number of data:" n
            "Average:" (/ sum n)
            "Minimum:" min
            "Maximum:" max
            "Range:" (- max min)))
```