York University- Department of Computer Science and Engineering

SC/CSE 3401 3.00 – Functional and Logic Programming

Assignment 4

<u>Attention!</u>

1- This assignment is due on Aug 2, 2010 at 14:30.

2- To submit this assignment, you need to email me (vida@cse.yorku.ca) two files as mentioned below. You will get a confirmation email to confirm receipt of your files. (If you don't get the email, it means I have not received your files.)

3- The first file must contain your code for this assignment. The filename should be "Yourlastname_Yourfirstname_code.lsp". Start your file with comment lines stating your name and student number. Include the answers to all questions in this file. Use comments to clearly separate each question and its parts. This file needs to be loadable into lisp.

4- The second file to submit is your dribble file. The filename should be "Yourlastname_Yourfirstname_dribble.txt". Use the dribble command to show your code works, at least for the examples mentioned in the questions. When done recording, add a few lines to the beginning of the file, stating your name and student number. If you need to write any explanations, add it to the beginning of this dribble file.

5- If you cannot write a perfect code, write a simpler, but working version to get a partial mark. If your code does not load or run at all (e.g. because of single parentheses), you will get a zero mark. Comment out any part of code that is syntactically wrong, if you are guessing!

6- You can only use the built-in functions mentioned in class for this assignment.

7- The efficiency and readability of your code is important. You will lose mark if you have an unnecessarily long code.

8- Check the website for updates, and Q & A.

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
(b) add2list2: use recursion
(c) add2list3: use mapcar

---

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).

(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
```

(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)).

(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
```

---

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.