York University- Department of Computer Science and Engineering

SC/CSE 3401 3.00 – Functional and Logic Programming

Assignment 2

---

1) This assignment is due on **February 24, 2012 <u>at 1:00pm in course drop box</u>**
2) Please provide your first name, last name and student number on the first page of your assignment.
3) Review policy on academic honesty. The submitted assignment must be each individual's own work.
4) NO LATE ASSIGNMENTS!
5) You need to **submit this assignment electronically. Include all your code in one single file. Comment your code, stating question number, plus a short explanation of the technique you used. For example 'A in this predicate is an accumulator that contains sum of numbers accumulated so far', etc.**
6) **You must not use any predefined code from Prolog, unless mentioned. This includes reversing and appending lists.**

---

1) (5 marks) Items in a list:
(a) Write noEven(L,N) which is true when N is the number of even numbers in L. For example:
:- noEven([2,5,7,8],N).
N = 2;
false.

(b) Write repCount(L, X, N) which is true when N is the number of occurrences of X in list L. For example:
:- repCount([5,2,3,5,2,4], 5, N).
N = 2;
false.

---

2)(10 marks) Digits of natural numbers:
(a) Write digitsR(N,L) which given a natural number N returns the digits of N in list L, starting from the least significant digit. For example:
:- digitsR(523, L).
L= [3, 2, 5].

(b) Use digitsR and an accumulator to write sumDigits(N, S) which given a natural number N returns the sum of its digits in S. For example:
:- sumDigits(523,S).
S = 10.
(c) Use difference lists to rewrite the code in part (a) as digitsD(N,L), returning the digits of N in list L starting from the most significant digit. Note the order of digits in L will change. For example:
:- digitsD(523,L).
L = [5, 2, 3].
(d) Use digitsD and difference lists to write digList(L, R) which given a list of natural numbers returns a list containing lists of digits. For example:
:- digList([523, 10, 12], R).
R = [[5, 2, 3], [1, 0], [1, 2]].

---

3)(10 marks) Depth of list items:
Write Prolog code for depthLevel(L,R) which given a nested list L returns R which is a list of pairs [E,D], where E is an element from L, and D is its depth in the list L. For example, for a query such as:
?- depthLevel([george, [bob,diana], [sue, [jack]], mary], R).
Prolog's response would be:
R = [[george, 1], [bob, 2], [diana, 2], [sue, 2], [jack, 3], [mary, 1]].

Hint: You can use difference lists for R (preserving the order of the elements) and an accumulator to keep track of depth so far. Keeping track of depth is similar to 'pretty print' we covered in class.

---

4) (10 marks) Number sequence:
Consider a sequence of numbers, S= 1, 1, 3, 7, 17, ... in which $a_n = a_{n-2} + 2* a_{n-1}$ and $a_0=1$, $a_1=1$.

(a) (5 marks) Use accumulators to write predicate *seq(A,B)* which given A will return B, where A and B are two consecutive numbers in this series. If A is not a member of the sequence, it must return fasle. For example:

?- seq(7,X).
X = 17.

?- seq(8,X).
false

(b) (5 marks) write seqN(A, N, L) which given A, A>1 and N, N>0, will return list L containing the next N numbers in above sequence. Your code is not required to work for A<=1. You can use reverse to re-order your list. Example:

?- seqN(3,5,L).

L = [7, 17, 41, 99, 239].

---

5) (12 points) Finding path to goal:

(a) (2 points) Write readmap(Filename, NoRows, NoCols, M) which read a map structure from file Filename and instantiates NoRows, NoCol, and M by reading from the file. For example file map1.txt contains a map of 4 rows and 5 columns). Therefore a query such as
?- readmap('map1.txt', NR,NC,M).
Would return:
NR = 4,
NC = 5,
M = map(0, 0, 0, x, 0, 0, 0, 0, 0, 0, 0, 0, 0, x, 0, 0, 0, 0, x, g).

(b) (2 points) Assume that each location on the map consists of a list [R,C] where R refers to the row and C refers to column. Write move(CurrentLocation, NewLocation, [NoRows, NoCol]) which allows moving right and down in the map. Note moving to left, up or diagonally is not a valid move. Use third argument [NoRows, NoCol] as a given map size to ensure the NewLocation is not outside the map's boundary. Therefore the following queries will be answered in the following way:

?- move([2,4],[R,C],[4,5]).
R = 2,
C = 5 ;
R = 3,
C = 4.

?- move([1,5],[R,C],[4,5]).
R = 2,
C = 5.

(c) (3 points) Write getPosMap(M, NR,NC, Loc, A) which returns A as the argument of M with NR rows and NC columns at Loc. Note Loc is list [R,C] containing row and column of the specified location. For example location [3,4] of map in map1.txt contains 'x', therefore the following query will match A to x.

?- readmap('map1.txt', NR, NC, M), getPosMap(M, NR, NC, [3,4], A).
NR = 4,
NC = 5,

M = map(0, 0, 0, x, 0, 0, 0, 0, 0, 0, 0, 0, x, 0, 0, 0, 0, x, g),
A = x.

(d) (5 points) Write pathList(FileName, Loc, R) which returns a path R starting from location Loc to a position in map containing 'g' , where the map is read from file FileName. The path must not consist of locations in map that have 'x' in them (they are the obstacles!). We will assume the goal is accessible by using moves to right and down. You can use **append** in your code**.**

You can expect two possible paths to the goal for the following query:

?- pathList('map1.txt', [1,2], R).
R = [[1, 2], [1, 3], [2, 3], [2, 4], [2, 5], [3, 5], [4, 5]] ;
R = [[1, 2], [2, 2], [2, 3], [2, 4], [2, 5], [3, 5], [4, 5]] ;
false.

Note added notes in first page!