COMP 3031 Assignment 2

Logic Programming

Fall 2016

Due: 5PM on Nov 30 Wednesday

Instructions

- There are five problems in this assignment. Each problem counts for two points.
- Write your prolog program according to the definition of the problem, with the same predicate name and number of arguments as specified. Write all the solutions in a single file named "ass3.pl".
- Submit your code through Canvas.
- **No** late submissions will be accepted.
- Your submission will be run on a lab 2 machine with the following command: "?- [ass3].".
  Please make sure your submission is executable. If it is not, a significant number of points will be deducted.

1. Dot product of integer lists

Define a predicate `dot(L1,L2,X)` to compute the dot product X of two integer lists $L1$ and $L2$. If $L1$ and $L2$ are of different lengths, the predicate will return false.

Examples:

```
?- dot([],[],X).
false.

?- dot( [1, 2], [1, 2], X).
X = 5.

?- dot( [1, 2, 3], [1, 2, 3], 14).
true.

?- dot( [1, 2, 3], [1, 2, 3], 1).
false.

?- dot( [1, 2, 3], [1, 2, 3, 4], X).
false.
```

2. Sliding window of integer list

Define a predicate `enum(X,N,L)`, where X is a list of integers, N is a non-negative integer, and L is a list consisting of all length-N sub-lists of X. In other words, each element of L is of the form $[X_i, X_{i+1}, \ldots, X_{i+N-1}]$, where X_i is the i-th element of X, and $i \in [1, len(X) - N + 1]$.

Examples:

```
?- enum([],2,L).
L = [].

?- enum([1, 2, 3, 4], 2, L).
L = [[1, 2], [2, 3], [3, 4]].

?- enum([1, 2, 3, 4], 5, L).
L = [].

?- enum([1, 2, 3, 4], 0, L).
L = [].

?- enum([1, 2, 3, 4], 1, [ [1], [2], [3]]).
false.

?- enum([1, 2, 3, 4], 1, [ [1], [2], [3], [4]]).
true.
```

For the next three problems, we define a relation `adj([v, [adj_v1, adj_v2, …, adj_vn]])` [updated Nov. 17, 2016], where `v` is a vertex in a directed graph, and `[adj_v1, adj_v2, …, adj_vn]` is the adjacency list of `v`, i.e., `(v, adj_v1), (v, adj_v2), …, (v, adj_vn)` are directed edges from `v` to `adj_v1, adj_v2, …, adj_vn`. The label of a vertex is alphanumeric. Assume each vertex in the graph is represented by one and only one `adj` fact in the database, and vertices in an adjacency list are distinct. Assume there is no self-loop on any vertex.

Examples are based on the following database:

```
/* The database of adj facts */
adj([a, [b,c,d]]).
adj([b, [d,e]]).
adj([c, []]).
adj([e, [a]]).
adj([d, [b]]).
```

3. List of the vertices in a graph

Define a relation `vlist (L, N)` that specifies a list `L` of N vertices in the graph in the order of the adj facts. N is less than or equal to the total number of vertices in the graph.

Examples:

```
?- vlist(L,0).
L = [].

?- vlist(L,1).
L = [a].

?- vlist(L,2).
L = [a, b].

?- vlist(L,3).
L = [a, b, c].

?- vlist(L,4).
L = [a, b, c, e].

?- vlist(L,5).
L = [a, b, c, e, d].

?- vlist([a,b,c],5).
false.

?- vlist([a,b,c],3).
true.

?- vlist([f, b, c, d, e], 5).
false.
```

4. Degree of a vertex

Write a relation `degree (V, D)` that specifies the out-degree (i.e., the number of out edges) of `V` in the graph. If V does not exist in the graph, the predicate returns false.

Examples:

```
?- degree(a, D).
D = 3.

?- degree(c, D).
```

```
D = 0.

?- degree(f, D).
false.

?- degree(X, D).
X = a,
D = 3 ;
X = b,
D = 2 ;
X = c,
D = 0 ;
X = e,
D = 1 ;
X = d,
D = 1.
```

5. Edge detection

Write a relation `edge(V1, V2)` that represents the edge `(V1, V2)` in the graph.

Examples:

```
?- edge(a, b).
true.

?- edge(b, a).
false.

?- edge(e, f).
false.

?- edge(a, X).
X = b ;
X = c ;
X = d ;
false.

?- edge(X, a).
X = e ;
false.

?- edge(X, Y).
X = a,
Y = b ;
```

```
X = a,
Y = c ;
X = a,
Y = d ;
X = b,
Y = d ;
X = b,
Y = e ;
X = e,
Y = a ;
X = d,
Y = b.
```