TDDD08 - Exercise 1

Valerio Colitta (950714-T639) valco263@student.liu.se
Théo Foray (980410-T096) thefo365@student.liu.se

September 2018

1 Exercise 1.1 - A Small Deductive Database

```
%Question 1
%beautiful(X) - X is beautiful
beautiful(ulrika).

%Question 2
%rich(X) - X is rich
beautiful(nisse).
rich(nisse).

%Question 3
%strong(X) - X is strong
rich(anne).
strong(anne).

%Question 4
strong(peter).
beautiful(peter).

%Question 5
%kind(X) - X is kind
kind(bosse).
strong(bosse).

%man(X) - X is a man
%likes(X,Y) - X likes Y
%woman(X) - X is a woman
man(peter).
man(bosse).
woman(anne).
woman(ulrika).
man(nisse).
```

```
%Question 6
likes(X,Y) :-man(X)
       ,beautiful(Y)
       ,woman(Y).
%query that illustrate this procedure : likes(X,ulrika).
%So we get all the men because they all like ulrika who's
%a beautiful woman.

%Question 7
%happy(X) - X is happy
happy(X):-rich(X).
%queries that illustrate this procedure:happy(nisse). happy(anne).

%Question 8
happy(X):-man(X),
       woman(Y),
       likes(X,Y),
       likes(Y,X).
%query that illustrate this procedure : happy(peter).
%Peter is the only one so far to be happy because he likes
%a woman who likes him (ulrika likes him because of Question 12).

%Question 9
happy(X):-man(Y),
       woman(X),
       likes(Y,X),
       likes(X,Y).
%query that illustrate this procedure : happy(ulrika).
%peter likes her and she likes peter (according to Question 12)
%so she's happy

%Question 10
likes(nisse,X):-woman(X),
       likes(X,nisse).
%query that illustrate this procedure :
%(likes(nisse,X). gives ulrika as an answer because she
% is a beautiful woman)

%Question 11
likes(peter,X):-kind(X).
%query that illustrate this procedure : likes(peter,bosse).

%Question 12
likes(ulrika,X):-man(X),
```

```
        rich(X),
        kind(X),
        likes(X,ulrika).

likes(ulrika,X):-man(X),
        beautiful(X),
        strong(X),
        likes(X,ulrika).
%query that illustrate this procedure : likes(ulrika,X).
%We only get peter because he likes her (she is a beautiful woman)
%and he is strong and beautiful.

%"Who is happy ?"
%| ?- happy(X).
%X = nisse ? ;
%X = anne ? ;
%X = peter ? ;
%X = ulrika ? ;
%no

%"Who likes who ?"
%| ?- likes(X,Y).
%X = peter,
%Y = ulrika ? ;
%X = nisse,
%Y = ulrika ? ;
%X = peter,
%Y = bosse ? ;
%X = ulrika,
%Y = peter ? ;
%no

%length(findall(),len).
%write(len).

count(Count):-findall(0, likes(ulrika,X), L),
        length(L, Count).
%To get how many persons like ulrika, this query gives the result:
%count(Count).

%In the procedures that need recursive calls, the recursive calls
%are done at the end, so that if the previous predicates are false
%we will avoid the infinite loops.
%We also must define the rules we are using in any other rule.
```

Listing 1: Ex 1.1

2 Exercise 1.2 - Recursion

```
e(a,b).
e(a,c).
e(b,c).
e(c,d).
e(d,h).
e(c,e).
e(d,f).
e(e,f).
e(e,g).
e(f,g).

%performing transitive closure for each node in the graph
path(X,Y) :-e(X,Y).
path(X,Y) :-e(X,Z), path(Z,Y).
%Usecase
%-----------------------------------------------------------------
%path(a,b). yes
%path(h,f). no
%-----------------------------------------------------------------

%path finder just like before, but now it gets recorded into a
%list.
%First, I record a two-length path into a list as soon as i find
%an edge.
%Then, given that I have a list for the path (Z,Y), I just need to
%append the next.
path(X,Y,[X,Y]) :-e(X,Y).
path(X,Y,[X|L]) :-e(X,Z), path(Z,Y,L).



%Usecases
%-----------------------------------------------------------------
%path(a,g,L). I want all the paths from a to g
%(if there are any paths)
%L = [a,b,c,d,f,g] ? ;
%L = [a,b,c,e,g] ? ;
%L = [a,b,c,e,f,g] ? ;
%L = [a,c,d,f,g] ? ;
%L = [a,c,e,g] ? ;
%L = [a,c,e,f,g] ? ;

%path(c,b,L). no, since c is not connected to b by any path.
```

```
%-----------------------------------------------------------------

%npath returns yes or no, depending on wheter there exists a
%path from X to Y. Moreover, if the call to path(X,Y,L) returns
%true, it calls the predicate length on the returned list,
%in order to get its length.
npath(X,Y,G) :-path(X,Y,L), length(L,G).
%Usecases
%-----------------------------------------------------------------
%npath(a,g,K).
%K = 6 ? ;
%K = 5 ? ;
%K = 6 ? ;
%K = 5 ? ;
%K = 4 ? ;
%K = 5 ? ;
%It reflects the previous test-query.

%npath(h,g,K). no. Since there is no path from h to g
%the second predicate (length) will not be called at all.
%-----------------------------------------------------------------
```

Listing 2: Ex 1.2