Declarative Programming and (Co)Induction
Module 2
Prolog lab 1

Davide Ancona and Elena Zucca
University of Genova

PhD Course, DIBRIS, June 26-27, 2014

1. Consider the following Prolog program:

```
:- use_module(library(coinduction)).

:- coinductive is_nat_co/1.

is_nat_co(z).
is_nat_co(s(N)) :- is_nat_co(N).

is_nat(z).
is_nat(s(N)) :- is_nat(N).
```

   (a) Find a ground term *t* for which both queries ?- is_nat(*t*) and ?- is_nat_co(*t*) succeed.

   (b) Find a ground term *t* for which both queries ?- is_nat(*t*) and ?- is_nat_co(*t*) fail.

   (c) Find a ground term *t* for which the query ?- is_nat(*t*) does not terminate, whereas ?- is_nat_co(*t*) succeeds.

   (d) Is there a ground term *t* for which the query ?- is_nat(*t*) succeeds, whereas ?- is_nat_co(*t*) fails?

2. (a) Extend the program in exercise 1 to define the two predicates

   is_nat_list/1 (inductive)
   is_nat_list_co/1 (coinductive)

   that succeed if the argument is a list of natural numbers (according to is_nat_co/1 predicate).

   (b) Repeat points (a) to (d) of exercise 1 for the two defined predicates.

3. Extend the program in exercise 1 to define the following predicates on natural numbers; for each kind of predicates, both the inductive and the coinductive version have to be considered;

```
pos/2   %% predicate ``positive''
geq/2   %% predicate ``greater than or equal''
leq/2   %% predicate ``less than or equal''
gth/2   %% predicate ``greater than''
lth/2   %% predicate ``less than''
eq/2    %% predicate ``equal to''
odd/1   %% predicate ``is odd''
even/1  %% predicate ``is even''
```

4. Extend the program in exercise 3 to define the following predicates on lists of natural numbers, ordered according to the standard lexicographical order; for each kind of predicates, both the inductive and the coinductive version have to be considered;

```
all_pos/1   %% predicate ``all list members are positive''
geq/2   %% predicate ``greater than or equal''
leq/2   %% predicate ``less than or equal''
gth/2   %% predicate ``greater than''
lth/2   %% predicate ``less than''
eq/2    %% predicate ``equal to''
```

5. Repeating decimals corresponding to rational numbers in the interval $[0, 1[$ can be represented by regular lists of digits.

   Define the predicate `eq/2` that checks if two repeating decimals are equal.

   Hint: recall that some numbers are not uniquely represented.