AIT Semantic and Declarative Technologies Course

Homework P2-2: Prolog lists and arithmetic

For each task write a Prolog predicate that corresponds to the specification provided as a comment. You may use the following built-in predicates (described in the lecture slides): `X = Y`, `Y is Expr` and arithmetic comparisons. **Do not use other built-in predicates**.

You can solve the tasks below without defining helper predicates. However, if you wish, you can define helper predicates. If you do so, please provide a head comment describing the expected behaviour of the helper predicate.

1. Incrementing all elements in a number list

```
% increment_list(+L0, ?L): L is a list of numbers obtained
% from L0 by incrementing each element by 1.
% You can assume that L0 is given and is a list of numbers.

| ?- increment_list([], L).
L = [] ? ; no
| ?- increment_list([1.2,5,2], L).
L = [2.2,6,3] ? ; no
```

2. Collecting all positive elements of a number list

```
% pos_elems(+L0, -L): L0 is a list of numbers. Given L0, return in L the
% list of all elements in L0 that are positive, in arbitrary order.

| ?- pos_elems([], L).
L = [] ? ; no
| ?- pos_elems([1,-2,-8,3,0,2,1], L).
L = [1,3,2,1] ? ; no
```

Important note: Remember that Prolog tries to use all clauses whose heads match a given goal to execute this goal. As the predicate `pos_elems` is expected to deliver a single answer, exactly one of clauses has to complete successfully for any goal invoking `pos_elems`.

3. Finding the last element of a list

```
% list_last(+List, ?V): List is a list whose last element is V.

| ?- list_last([], V).
no
| ?- list_last([5,1,2,8,7], V).
V = 7 ? ; no
| ?- list_last([5,1,2], 2).
yes
```

4. Finding a value in a list

```
% list_element(+List, +V): V is present as an element in the list List.

| ?- list_element([], x).
no
| ?- list_element([a,b,c], a).
yes
| ?- list_element([d,c,1,2], 12).
no
```

You can assume that the list elements are all different. Please do **not** make any efforts to optimize the code by avoiding further search when the given element is found, as this will hinder the extension described in the next paragraph.

Ensure that your code also works for the input-output mode `list_element(+List, -V)`, by enumerating in `V` all list elements if `V` is a variable. (The order of the solutions does not matter.)

```
| ?- list_element([d,c,1,2], X).
X = d ? ; X = c ? ; X = 1 ? ; X = 2 ? ; no
```

5. Concatenating lists

The concatenation of lists [x_1, ..., x_k] and [y_1, ..., y_l] is the list [x_1, ..., x_k, y_1, ..., y_l].

```
% list_concat(+L1, +L2, -L3): Return in L3 the concatenation of given lists L1 and L2.

| ?- list_concat([], [a,1,x], L).
L = [a,1,x] ? ;
no
| ?- list_concat([p,q], [a,1,x], L).
L = [p,q,a,1,x] ? ;
no
```