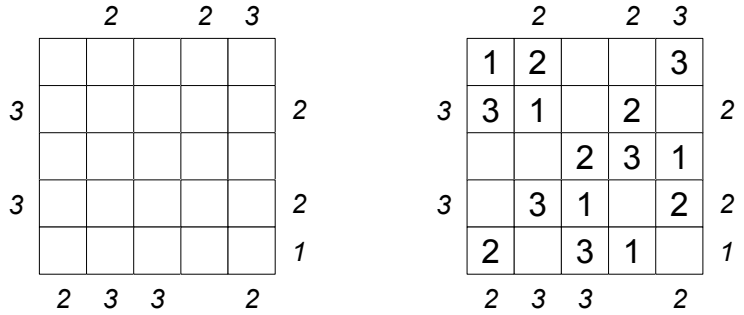
AIT Semantic and Declarative Technologies course
Final test/CLP part: The End View Puzzle

The End View Puzzle

A square board of size $n \times n$ is given. Numbers from the range $[1, m]$, $m \leq n$ are to be placed on the board, so that all integers in $[1, m]$ occur exactly once in each row and in each column (and there are $n - m$ empty fields in each of these). As clues, numbers are placed outside the board, prescribing the first number seen from the given direction. (Empty squares are assumed to be transparent.)

A sample puzzle of this kind is shown below, for $n = 5$, $m = 3$, together with its unique solution. Here, for example, the number 2 placed below the fifth column prescribes that the last number placed in that column is 2, while the number 3 placed in front of row 4 requires that this number is the first in that row.



Your task

Your task is not to solve such a puzzle, but to write an auxiliary predicate `endviews/5`, which could be used by a program that solves the End View Puzzle. The specification of `endviews/5` is as follows:

```
% endviews(+N, +M, ?List, +LV, +RV):
% The list List is of length N, and contains all numbers between 1 and M exactly once, and
% (N-M) instances of the number 0. LV (left view) is the first nonzero element in the list
% List and RV (right view) is the last nonzero element. All arguments, except List, are given
% integers, where N>=M, 1 =< LV =< M and 1 =< RV =< M also holds. Furthermore, LV \= RV
% holds unless M = 1. List is a variable or a proper list of FD variables/integers.
```

You don't need to check the conditions on the four input arguments listed above, but you may assume that these conditions hold for all test cases.

The predicate `endviews/5` should **not** create any choice points, nor should it call any labeling predicates. The first test case below illustrates this. In this test case your program should produce just a single answer. If pressing ';' after the first answer produces a second one, this is a sign that there is a choice point in your code, which is not acceptable.

In the submission test, the very first test case checks whether your code creates a choice point and produces multiple answers. Be warned that if the first test case fails for this reason, your solution cannot be accepted, even if all the remaining ones are successful.

Try to make the constraint as strong as possible (i.e., as near to being domain consistent as possible). Your program may provide some less strict answers than the ones shown on the next page, as long as it produces the prescribed set of answers for examples with labeling (the order of the solutions can be different from the one on the next page).

The last four test cases – which are similar to the last four examples overleaf – are optional. These test cases contain no labeling, and measure the strength of pruning. You can get 12 optional points if your program does strong pruning for these inputs. (It is suggested that you first come up with a solution that passes the test cases with labeling, before embarking on this more ambitious task.)

Hints

Use the `global_cardinality` constraint.
Reduce the endviews problem to two instances of a "left view" constraint.
In implementing the latter, use the aproach similar to the `nsum` constraint shown in the lecture slides:

- first write a Prolog solution,
- next (still in Prolog) rewrite it to (unnecessarily) scan the whole list,
- finally transform this solution to use `library(clpfd)`.

Example runs

```
| ?- endviews(3, 2, L, 1, 2).
L = [_A,_B,_C],
_A in 0..2, _B in 0..2, _C in 0..2 ? ;
no
| ?- endviews(3, 2, L, 1, 2), labeling([], L).
L = [0,1,2] ? ; L = [1,0,2] ? ; L = [1,2,0] ? ;
no
| ?- endviews(7, 5, L, 2, 3), L = [0,_,_,1,4,_,_], labeling([], L).
L = [0,0,2,1,4,5,3] ? ;
L = [0,2,0,1,4,5,3] ? ;
L = [0,2,5,1,4,0,3] ? ;
L = [0,2,5,1,4,3,0] ? ;
no
| ?- endviews(7, 5, L, 2, 3), L = [0,_,_,1,_,_,_], labeling([], L).
L = [0,0,2,1,4,5,3] ? ;
L = [0,0,2,1,5,4,3] ? ;
L = [0,2,0,1,4,5,3] ? ;
L = [0,2,0,1,5,4,3] ? ;
L = [0,2,4,1,0,5,3] ? ;
L = [0,2,4,1,5,0,3] ? ;
L = [0,2,4,1,5,3,0] ? ;
L = [0,2,5,1,0,4,3] ? ;
L = [0,2,5,1,4,0,3] ? ;
L = [0,2,5,1,4,3,0] ? ;
no
| ?- endviews(7, 5, L, 2, 3).
L = [_A,_B,_C,_D,_E,_F,_G],
_A in{0}\/{2},
_B in(0..2)\/(4..5),
_C in(0..2)\/(4..5),
_D in(0..1)\/(4..5),
_E in(0..1)\/(3..5),
_F in(0..1)\/(3..5),
_G in{0}\/{3} ? ;
no
| ?- endviews(7, 5, L, 2, 3), L = [0|_].
L = [0,_A,_B,_C,_D,_E,_F],
_A in{0}\/{2},
_B in(0..2)\/(4..5),
_C in(0..1)\/(4..5),
_D in(0..1)\/(4..5),
_E in(0..1)\/(3..5),
_F in{0}\/{3} ? ;
no
| ?- endviews(7, 5, L, 2, 3), L = [0,_,_,1,_,_,_].
L = [0,_A,_B,1,_C,_D,_E],
_A in{0}\/{2},
_B in{0}\/{2}\/(4..5),
_C in{0}\/(4..5),
_D in{0}\/(3..5),
_E in{0}\/{3} ? ;
no
| ?- endviews(7, 5, L, 2, 3), L = [0,_,_,1,4,_,_].
L = [0,_A,_B,1,4,_C,_D],
_A in{0}\/{2},
_B in{0}\/{2}\/{5},
_C in{0}\/{3}\/{5},
_D in{0}\/{3} ? ;
no
```