Santa always leaves plans for his elves to determine the order in which the reindeer will pull his sleigh. This year, for the European leg of his journey, his elves are working to the following schedule, which will form a single line of nine reindeer.

Here are the rules:

- Comet behind Rudolph, Prancer and Cupid
- Blitzen behind Cupid
- Blitzen in front of Donder, Vixen and Dancer
- Cupid in front of Comet, Blitzen and Vixen
- Donder behind Vixen, Dasher and Prancer
- Rudolph behind Prancer
- Rudolph in front of Donder, Dancer and Dasher
- Vixen in front of Dancer and Comet
- Dancer behind Donder, Rudolph and Blitzen
- Prancer in front of Cupid, Donder and Blitzen
- Dasher behind Prancer
- Dasher in front of Vixen, Dancer and Blitzen
- Donder behind Comet and Cupid
- Cupid in front of Rudolph and Dancer
- Vixen behind Rudolph, Prancer and Dasher.

Create a decision model that determines the order of the reindeer and send your solution to DecisionManagementCommunity@gmail.com.

Note

* All 15 constraints are not needed. In fact only 9 constraints are needed in order to solve the problem without search. Below I’ve comment out the one that’s not needed.
* I’m not sure if this is the only combination of 9 constraints that is needed. Perhaps there is some other combination of 9 constraints that will suffice.

This MiniZinc model was created by Hakan Kjellerstrand, hakank@gmail.com
See also my MiniZinc page: http://www.hakank.org/minizinc/

```
include "globals.mzn";

enum reindeer = {blitzen,comet,cupid,dancer,dasher,donder,prancer,rudolph,vixen};

% decision variables
var reindeer: Blitzen;
var reindeer: Comet;
var reindeer: Cupid;
var reindeer: Dancer;
var reindeer: Dasher;
var reindeer: Donder;
var reindeer: Prancer;
var reindeer: Rudolph;
```
var reindeer: Vixen;

% places of each of the reindeer
array[reindeer] of var reindeer: x =
   [Blitzen, Comet, Cupid, Dancer, Dasher, Donder, Prancer, Rudolph, Vixen];
% the order of the reindeer (the solution)
array[reindeer] of var reindeer: order;

% x is behind all elements in y
predicate behind(var int: x, var set of int: y) = forall(i in y) ( x > i );

% x is in front of all elements in y
predicate in_front_of(var int: x, var set of int: y) = forall(i in y) ( x < i );
solve satisfy;

costRAINT
   all_different(x) /\
   inverse(x, order) /\

   % Comet 'behind' {Rudolph, Prancer, Cupid} /\
   % Blitzen 'behind' {Cupid} /\
   Blitzen 'in_front_of' {Donder, Vixen, Dancer} /\
   % Cupid 'in_front_of' {Comet, Blitzen, Vixen} /\
   Donder 'behind' {Vixen, Dasher, Prancer} /\
   % Rudolph 'behind' {Prancer} /\
   Rudolph 'in_front_of' {Donder, Dancer, Dasher} /\
   Vixen 'in_front_of' {Dancer, Comet} /\
   Dancer 'behind' {Donder, Rudolph, Blitzen} /\
   Prancer 'in_front_of' {Cupid, Donder, Blitzen} /\
   % Dasher 'behind' {Prancer} /\
   Dasher 'in_front_of' {Vixen, Dancer, Blitzen} /\
   Donder 'behind' {Comet, Cupid} /\
   Cupid 'in_front_of' {Rudolph, Dancer} % /\
   % Vixen 'behind' {Rudolph, Prancer, Dasher}
;

output ['\(order)\n'];