element(Index, List, Elem) List[Index] = Elem

global_cardinality(List, Values) [1-3, 2-1, 3-Y, 4-X], (Y #= 2 #/\ X #=3) #\/ (Y #= 4 #/\ X#=

sum(List, RelOp, Value) sum(List, #= , 5), sum(List, #<, X), X...
#= #<

scalar_product(Coeffs, List, RelOp, Value)

Constraint #<=> Val

F	0
V	1

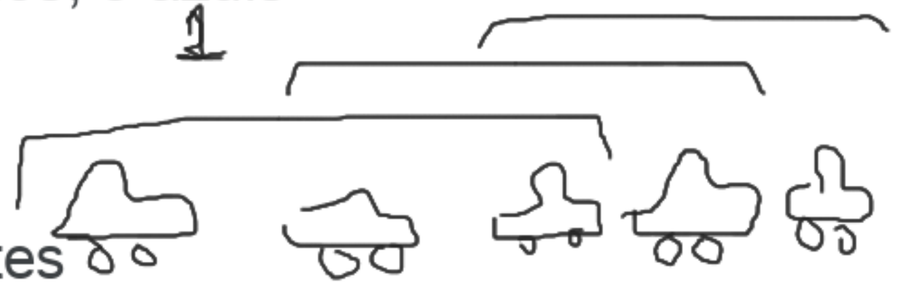
count_equals(Val, List, Count)

```
count_equals(_, [], 0).
count_equals(Val, [H|T], Count) :-
    Val #= H #<=> B,
    Count #= Count1 + B,
    count_equals(Val, T, Count1).
```

Count_Final #= B_1 + B_2 + B_3 + ... + 0

12 carros estão parados, em fila, num semáforo. Sabe-se que :

- A distribuição de cores é: 4 amarelos, 2 verdes, 3 vermelhos, 3 azuis
- O primeiro e último têm a mesma cor
- O segundo e o penúltimo têm a mesma cor
- O quinto é azul
- Todas as sub-sequências de 3 carros têm 3 cores diferentes
- Do primeiro para o último, a sequência amarelo-verde-vermelho-azul aparece uma única vez



```
carros:-
    length(List,12),
    global_cardinality(List, [2-4, 3-2, 4-3, 1-3]),
    element(1, List, X), element(12, List, X),
    element(2, List, Y), element(11, List, Y),
    element(5, List, 1),
    three_diferent(List),
    only_once(List, 1),
    labeling( [ ], List),
    write(List).
```

```
only_once(List, 0):-
    length(List, L),   L < 4.
only_once([A,B,C,D|T], Counter):-
    (A #= 2 #/\ B #=3 #/\ C #=4 #/\ D #= 1) #<=> Bin,
    Counter #= Counter1 + Bin,
    only_once([B,C,D|T],Counter1)
```

```
% List = [A, B, C, D, 1, F, G, H, I, J, B, A],
```

```
three_diferent(List) :-
  length(List, L),
  L < 3.
three_diferent([A, B, C|T]) :-
  all_distinct([A, B, C]),
  three_diferent([B,C|T]).
```

1 - azul
2 - amarelo
3 - verde
4 - preto

carros:-

```
Cor = [ A, B, C, D ],    %posição = posição na fila; valor é cor
Tam = [ E, F, G, H ],   %
domain(Cor, 1, 4), domain(Tam, 1, 4), all_distinct(Cor), all_distinct(Tam),
element(Index, Cor, 3), element(Index, Tam, 1),  % verde é menor

element(I, Cor, 1), IndAntesAz #= I+1, IndDepoisAz #= I-1,              % I é index do azul
element(IndDepoisAz, Tam, TamDepois), element(IndAntesAz, Tam, TamAntes),
TamAntes #< TamDepois,                          %imed antes azul menor imed dep azul

Index #< I,       %verde depois do azul
element(Yellow, Cor, 2), element(Black, Cor, 4), Yellow #< Black,    %amarelo dep. preto
append(Cor, Tam, Vars),
labeling([ ], Vars).
```

Parabéns, chegou ao fim
Força neste resto de semestre camarada