Declarative Programming with Constraints – Homework practice C3

For each of the following problems, write a Prolog program – using the SICStus CLP(FD) library – that corresponds to the specification provided as a comment. Try to make your program efficient.

1. In a list of numbers, an element is a *local optimum* if it is either greater than both of its neighbors or it is less than both of its neighbors. (The first and last elements of the list are not considered to be local optima.)

```
% optnum(+L, ?K): the number of local optima in list L is K and the elements of L are all distinct.
% It can be assumed that L is proper (i.e., not open-ended). The elements of L as well as K are
% FD-variables or integers. optnum/2 should not perform labeling nor create any choice points.
```

```
| ?- L=[1,_,_,_], domain(L, 1, 4), optnum(L, 2), labeling([], L).
L = [1,3,2,4] ? ;
L = [1,4,2,3] ? ;
no
```

2. In a list of integers, an element is *visible from the left* if it is greater than all the preceding elements of the list. (The first element of the list is always visible from the left.)

```
% visnum(+L, ?K): L is a list of positive integers.
%                 The number of elements in L that are visible from the left is K.
% It can be assumed that L is proper (i.e., not open-ended). The elements of L as well as K are
% FD-variables or integers. visnum/2 should not perform labeling nor create any choice points.
```

```
| ?- L=[_,_,2,_], domain(L, 1, 4), all_distinct(L), visnum(L, 3), labeling([], L).
L = [1,3,2,4] ? ;
no
```

3. Global constraints are not reifiable. In some cases, it is useful to manually implement the reified version of a global constraint.

```
% alldifferent_reif(+L, ?B): B is a boolean value (0 or 1).
% The elements of list L are all pairwise different if and only if B=1.
% It can be assumed that L is proper (i.e., not open-ended).
% B and the elements of L are FD-variables or integers.
% Predicate alldifferent_reif/2 should not perform labeling nor create any choice points.
```

```
| ?- alldifferent_reif([4,5,6,5,7],B).
B = 0 ? ;
no
| ?- alldifferent_reif([4,50,7],B).
B = 1 ? ;
no
| ?- alldifferent_reif([4,X],0).
X = 4 ? ;
no
| ?- alldifferent_reif([4,X,5],1).
X in(inf..3)\/(6..sup) ? ;
no
| ?- X in 0..3, Y in 6..11, alldifferent_reif([X,Y,5],B).
B = 1, X in 0..3, Y in 6..11 ? ;
no
| ?- X in 0..20, Y in {3,6}, alldifferent_reif([4,Y,X],0), labeling([], [X,Y]).
X = 3, Y = 3 ? ;
X = 4, Y = 3 ? ;
X = 4, Y = 6 ? ;
X = 6, Y = 6 ? ;
no
```

Hints:

- To understand what is expected from the above predicate, it may help to first try implementing variants that work for some simple special cases. For example:
  `alldifferent_reif([X,Y], B) :- X #\= Y #<=> B.`
  Try writing the body of the predicate below. Use reification and propositional constraints.
  `alldifferent_reif([X,Y,Z], B) :- ... .`
  Note that working out the special case for three variables serves only for understanding the general case with n variables. The special case is **not** envisaged to be used as a helper.
- You may consider using predicate `pairings/2` from Homework P3.