Les nombres premiers

- But:
  - trouver tous les nombres premiers de 1 à N
- Crible d'Ératosthène:
  1. Générer tous les entiers de 2 à N
  2. Supprimer tous les multiples de 2, de 3, de 4, etc.
  3. Arrêter lorsque le carré du plus petit entier non traité est supérieur au grand entier de la liste.

Les nombres premiers

Seules les règles 1. et 2. du crible d'Ératosthène sont appliquées ici

```
genereListe(0,[]).
genereListe(N,[N|Xs]):- N > 0,
                N1 is N-1,genereListe(N1,Xs).

retireMultiple(X,[],[]).
retireMultiple(X,[T|Q],Resultat) :- T>X,
   T mod X =:= 0, retireMultiple(X,Q,Resultat),!.
retireMultiple(X,[T|Q],[T|Resultat]) :-
   retireMultiple(X,Q,Resultat).

retireTousLesMultiples(N,[],[]).
retireTousLesMultiples(1,L,L).
retireTousLesMultiples(N,Li,L):- N>1, retireMultiple(N,Li,LL),
                N1 is N-1, retireTousLesMultiples(N1,LL,L).

premiers(N,L):- genereListe(N,Li),
            retireTousLesMultiples(N,Li,L).
```

Exemple du fermier

% etat(Fermier,Renard,Poule,Blé).

initial(etat([gauche, gauche, gauche, gauche])).
final(etat([droite, droite, droite, droite])).

Exemple du fermier

traverse(etat([gauche,X,Y,Z]),etat([droite,X,Y,Z]), fermier_traverse).
traverse(etat([droite,X,Y,Z]),etat([gauche,X,Y,Z]), fermier_revient).

traverse(etat([gauche,X,gauche,Z]),etat([droite,X,droite,Z]), fermier_amene_poule).
traverse(etat([droite,X,droite,Z]),etat([gauche,X,gauche,Z]), fermier_ramene_poule).

traverse(etat([gauche, gauche, X, Y]),etat([droite, droite, X, Y]), fermier_amene_renard).
traverse(etat([droite, droite, X, Y]),etat([gauche, gauche, X, Y]), fermier_ramene_renard).

traverse(etat([gauche, X, Y, gauche]),etat([droite, X, Y, droite]), fermier_amene_ble).
traverse(etat([droite, X, Y, droite]),etat([gauche, X, Y, gauche]), fermier_ramene_ble).

Exemple du fermier

```
interdit(etat([X, Y, Y, _])) :- X \== Y.
interdit(etat([X, _, Y, Y])) :- X \== Y.

riviere(P) :-
    initial(Depart), final(Arrivee),
    riviere_aux(Depart, Arrivee, [Depart], P).
```

Exemple du fermier

```
riviere_aux(A,A,_,[]).

% V sont les états déjà visitées
riviere_aux(A, B,V, P) :-
    traverse(A,C,Action),
    not(interdit(C)),
    not(member(C,V)),
    riviere_aux(C,B,[C|V],Plan),
    P = [Action | Plan].
```

Le prédicat setof

```
aime(jean,pomme).
aime(marie,poire).

?- setof([X,Y],aime(X,Y),L).
L=[[jean,pomme],[marie,poire]].
```

```
age(pierre,5).
age(paul,7).
age(henri,5).

?- setof(C,age(C,5),L).
L=[henri,pierre].
```

bagof est similaire, sauf qu'il n'élimine pas les répétitions et ne tri pas les éléments.

Exemple

```
bag(2,4,1).
bag(3,5,2).
bag(7,8,2).
bag(4,3,1).
bag(5,2,4).
bag(2,1,4).
bag(2,2,4).
bag(7,3,5).
bag(7,3,3).
```

```
% bagof(Z,bag(X,Y,Z),B).
% bagof(Z,(bag(X,Y,Z),Z>2),B).
% bagof(Z,X^bag(X,Y,Z),B).
% setof(Z,X^bag(X,Y,Z),B).
% bagof(Z,X^Y^bag(X,Y,Z),B).
% findall(Z,bag(X,Y,Z),B).
```

Exemple

```
connait(vincent,david).
connait(vincent,antoine).
connait(vincent,alex).
connait(melodie,alex).
connait(melodie,patrick).
connait(patrick,melodie).
connait(patrick,ahmed).
connait(patrick,eddie).
connait(patrick,david).
```

```
% setof(X,connait(X,Y),B).
% setof(Y,connait(X,Y),B).
% setof(Y,X^connait(X,Y),B).
% bagof(Y,X^connait(X,Y),B).
% setof([X,Y],connait(X,Y),B).
```

Exemple

```
age(vincent,8).
age(melodie,4).
age(patrick,3).
age(ahmed,7).
age(eddie,4).
% setof(A,N^age(N,A),B).
% setof(A,N^age(N,A),[T|Q]).
% setof(A,N^age(N,A),[T|_]).
% setof([A,N],age(N,A),[[_,J]|_]).
% age(P,A1),\+((age(_,A2),A2<A1)).
```