```
% [mortel] : compréhension et assertion de phrases de la forme
% X est un Y ou tout X est un Y. Cf CC n3, 96/97.

pp :-
    repeat,
      lire(Phrase),
      decoup(Phrase,[],LAtomes),
      p(Clause,LAtomes,[]),
      traite(Clause),
    Clause = fin,
    !.

% Prédicat de lecture caractère par caractère
% La fin de la lecture est marquée par '.' (ascii 46)

lire(L) :-
    get0(Car),
    lire_reste(Car,L).

lire_reste(46,[]) :- !.
lire_reste(Car,[Car|Q]) :-
    get0(NCar),
    lire_reste(NCar,Q).

% Découpage d'une liste de caractères selon les blancs
% decoup(Liste de car, Liste restante, Liste d'atomes)
% Hypothèse simplificatrice : il y a un seul blanc pour séparer les
% mots.

decoup([],SL,[At]) :-
        name(At,SL).
decoup([32|R],SL,[At|La]) :-
        !,
        name(At,SL),
        decoup(R,[],La).
decoup([C|Q],L,La) :-
        append(L,[C],Lr),
        decoup(Q,Lr,La).
```

Université Paris 7 – LI411c – 96/97 Corrigé CC n° 3

```
% Grammaire DCG du fragment à traiter.
% On mélange des terminaux et des non terminaux
% Les représentations construites sont :
%       N(C)         pour << C est un N >>
%       Q(X) :- T(X) pour << tout T est un Q >>
%       ?- N(C)      pour << est ce que C est un N >>

p(L)          --> cte(X), [est], art, n(X,L).
p((Q :- T))   --> [tout], n(X,T), [est], art, n(X,Q).
p((?- T))     --> [est], [ce], [que], cte(X), [est], art, n(X,T).
% Juste pour simplifier l'interface.
p(fin)        --> [stop].

% Noter le "truc" pour produire un terme de la forme "homme(X)", avec
% possibilité d'instancier ensuite X.
n(X,homme(X)) --> [homme].
n(X,femme(X)) --> [femme].
n(X,individu(X)) --> [individu].
n(X,fermier(X)) --> [fermier].
n(X,coquette(X)) --> [coquette].

art --> [un].
art --> [une].
cte(jean) --> [jean].
cte(marie) --> [marie].
cte(paul) --> [paul].
cte(X) --> [X].

% Le traitement apporté aux représentations dépend de leur forme.
% faits : assertés
% règles : assertés
% questions : démontrées
% "fin" : ignoré

traite((?-T)) :-
    !, verif(T).
traite(stop) :-
    !.
traite(T) :-
    !, assert(T).

% Démontrer une question revient à appeler le prédicat correspondant.

verif(C) :- call(C), !, write('OK'), nl.
verif(_) :- write('Eh non !'), nl.
```