Prolog - TD ½

Listes :

```
tete([T|Q],T).
queue([T|Q],Q).
element1(E,[E|L]).
element1(E,[_|L]):- element1(E,L).
element2(E,L):- conc(L1,[E|L2],L).
hors_de(X,L):-
       nonvar(X),
       hors(X,L).
hors(X,[]).
hors(X,[Y|L]):-
       X \== Y,               /* ne marche que si X est connu: c'est pour  */
       hors_de(X,L).          /* cela que ce predicat n'est pas inversible */
conc([],L,L).
conc([T|L1],L2,[T|L3]):- conc(L1,L2,L3).
appartient(Sous_liste,Sur_liste):-
       conc(L2,L3,Sur_liste),
       conc(L1,Sous_liste,L2).
retrait_elt1(E,Ldepart,Lresultat):-
       conc(L1,[E|L2],Ldepart),
       conc(L1,L2,Lresultat).
retrait_elt2(J,[J|L],L).
retrait_elt2(J,[K|L],[K|M]):- retrait_elt2(J,L,M).
retrait_tous_les_elts(J,[J|L],LR):-
        retrait_les_elts1(J,L,LR).
retrait_tous_les_elts(J,[K|L],[K|M]):-
        K \== J,
       retrait_tous_les_elts(J,L,M).
retrait_les_elts1(_,[],[]).
retrait_les_elts1(J,[J|L],LR):-
        retrait_les_elts1(J,L,LR).
retrait_les_elts1(J,[K|L],[K|LR]):-
        K \== J,
        retrait_les_elts1(J,L,LR).
retrait_liste(Sous_liste,ListeInitiale,ListeResultat):-
       conc(L2,L3,ListeInitiale),
       conc(L1,Sous_liste,L2),
       conc(L1,L3,ListeResultat).
permuter([],[]).
permuter(L,[T|Q]):-
       retrait_elt2(T,L,L2),
       permuter(L2,Q).
renverser_1([],[]).
renverser_1([A|L],Z):-
       renverser_1(L,L1),
       conc(L1,[A],Z).
renverser_2(L1,L2):- renverser_3(L1,[],L2).
renverser_3([],L,L).
renverser_3([T|Q],L_aux,L):-
       renverser_3(Q,[T|L_aux],L).
```

Famille :

```
/* pere(pere,enfant) */
pere(pierre,francois).
pere(pierre,jean).
pere(andre,rose).
pere(jean,bastien).
pere(jean,rudolph).
pere(rudolph,gwendoline).
/* mere(mere,enfant) */
mere(mathilde,francois).
mere(mathilde,rose).
mere(rose,bastien).
mere(rose,rudolph).
parent(Parent,Enfant) :- mere(Parent,Enfant).
parent(Parent,Enfant) :- pere(Parent,Enfant).
descendant(Descendant,Ascendant) :-
  parent(Ascendant,Descendant).
descendant(Descendant,Ascendant) :-
  parent(Ascendant,Enfant),
  descendant(Descendant,Enfant).
```

Tri Naif :

```
:-[listes]. % Consulte le fichier listes.pl
 /*          LE TRI NAIF             */
tri(L1,L2):-
       permuter(L1,L2),
       ordonnee(L2).
```

```
ordonnee([]).
ordonnee([X]). /* Une liste a un element */
ordonnee([X,Y|L]):-
       X =< Y,
       ordonnee([Y|L]).
/*
| ?-  tri([1,2,6,3,9,4],L).
L = [1,2,3,4,6,9] ? ;
no
| ?-
*/
 /*          LE TRI PAR SELECTION            */
tri_s([H|L],[M|L1]):-
        extract_min(L,H,L2,M),
        tri_s(L2,L1).
tri_s([],[]).
extract_min([],M,[],M).
extract_min([H|L1],M0,[H|L2],M):-
        M0 =< H,
        extract_min(L1,M0,L2,M).
extract_min([H|L1],M0,[M0|L2],M):-
        H < M0,
        extract_min(L1,H,L2,M).
```

Zebre :

```
/* Solution Prolog
   on suppose que les maisons sont numerotees de 1 a 5 de gauche a droite.
   C est une liste de 5 couleurs [C1,C2,C3,C4,C5]
         tq Ci=couleur de la maison i
   N est une liste de 5 nationalites [N1,N2,N3,N4,N5]
         tq Ni=nationalite de l'habitant de la maison i
   B est une liste de 5 boissons [B1,B2,B3,B4,B5]
         tq Bi=boisson de l'habitant de la maison i
   A est une liste de 5 animaux [A1,A2,A3,A4,A5]
         tq Ai=animal de la maison i
   F est une liste de 5 marques de cigarettes [F1,F2,F3,F4,F5]
         tq Fi=marque des cigarettes fumees par l'habitant de la maison i
*/
/* Calcul des reponses */
   zebre(ProprietaireZebre,BuveurEau) :-
        relation(C,N,F,B,A),
        meme_position(zebre,ProprietaireZebre,A,N),
        meme_position(eau,BuveurEau,B,N).
/* Construction des listes a partir de l'information fournie */
relation(C,N,F,B,A):-
           C=[_,bleu,_,_,_], N =[norvegien,_,_,_,_],
           B= [_,_,lait,_,_], A= [_,_,_,_,_],
           meme_position(rouge,anglais,C,N),
           meme_position(vert,cafe,C,B),
           meme_position(jaune,kool,C,F),
           voisin_droit(vert,blanc,C),
           meme_position(espagnol,chien,N,A),
           meme_position(ukrainien,the,N,B),
           meme_position(japonais,craven,N,F),
           meme_position(old_gold,escargot,F,A),
           meme_position(gitane,vin,F,B),
           voisin(chesterfield,renard,F,A),
           voisin(kool,cheval,F,A).
/* meme_position(I,J,K,L). */
/* Les elements I et J sont a la meme position dans les listes K et L */
meme_position(I,J,[I|U],[J|V]).
meme_position(I,J,[M|U],[N|V]):-
           I \== M, % Optmisation possible car tous les elements doivent
           J \== N, % etre differents
           meme_position(I,J,U,V).
/* I et J occupent des positions voisines sur les listes U et V */
voisin(I,J,L1,[H|L2]):-
           H \== J,
           meme_position(I,J,L1,L2).
voisin(I,J,[H|L1],L2):-
           H \== I,
           meme_position(I,J,L1,L2).
/* voisin_droit(I,J,L) */
/* J est le voisin droit de I sur la liste L */
voisin_droit(I,J,L):-
           L=[H|L1],
           H \== J,
           meme_position(I,J,L,L1).
```