Dans l'ordre : domains / predicates / clauses / goal

predicates :
- minuscule(symbol/integer ou var de domains)
aimePas(symbol,symbol)

clauses :
-minuscules(X) :- condition
aime(bruno,pizza).

aime(L,pizza) renvoie toute les personnes qui aime les pizzas
prixLogementUneSemaine(L,_,_). renvoie toutes les destination
possibilité de faire un calcul
not(aime(paul,pizza)) signifie que l'on regarde si paul n'aime pas les pizzas

instruction dans le goal :
clearwindow / write(), readln(), nl, readint()

clauses importante :
factoriel
ValeurFactoriel(0,1).
ValeurFactoriel(1,1).
ValeurFactoriel(X, Result):- X > 1, X1 = X-1, ValeurFactoriel(X1, Result1), Result = Result1 * X.

taille d'une liste
tailleListe([],0).
tailleListe([_|Y],L):-tailleListe(Y,L2), L = L2+1.

Affichage d'une liste
affichageVille([]).
affichageVille([V|Q]):-write(V," "), affichageVille(Q).

findAll(X, distanceAvion(X,_),L) permet de recuperer toute les destinations possible et de les stocker dans une liste.

Concatener deux listes
concatenerListe([],L,L).
concatenerListe([X|Y],L,[X|R]):-concatenerListe(Y,L,R).
La derniere liste est celle vide au départ mais le résultat a la fin.
findAll(X, distanceAvion(X,_),L),
findAll(X2, distanceTrain(X2,_),L2),
concatenerListe(L,L2,L3),

Contenir
contient([V|_],X):-T=X.
contient([V|Q],X):-V<>X,contient(Q,X).

Le fail lui permet de ne pas s'arreter qd une instruction est vrai pratique dans le cas de recherche de deux valeurs d'une liste ;.
chercherSiDesservie(L1,_,X):-contient(L1,X),write(X," est desservie en train"), fail.
chercherSiDesservie(_,L2,X):-contient(L2,X),write(X," est desservie en avion").

Tjrs penser au not() pour savoir si un élément existe ou pas cf contenir.
chercherSiDesservie(L1,L2,X):-not(contient(L1,X)),not(contient(L2,X)),write(X," n'est pas desservie").

```
domains
        nom = symbol
        nbSelection = integer
        postePossible = symbol
        listeJ = nom*

predicates
        joueur(nom,nbSelection)
        poste(nom,postePossible)
        affichageDetail(nom)
        listeJparPoste(postePossible)
        affichageListeJ(listeJ)
        listeMc(listeJ)
        listeMd(listeJ)
        listeMg(listeJ)
        concatenerListe(listeJ,listeJ,listeJ)
        contenir(listeJ,nom)
        test(listeJ,nom)
        calculSomme(listeJ,integer)
        tailleListe(listeJ,integer)
        calculMoy(integer,integer,integer)

clauses
        joueur(henry,8).
        joueur(trezeguet,5).
        joueur(blanc,9).
        joueur(thuram,6).
        joueur(desailly,8).
        joueur(djorkaeff,10).
        joueur(lizarazu,10).
        poste(henry,avg).
        poste(henry,avd).
        poste(trezeguet,avc).
        poste(trezeguet,avd).
        poste(blanc,arc).
        poste(thuram,ard).
        poste(desailly,mg).
        poste(desailly,mc).
        poste(desailly,md).
        poste(djorkaeff,mc).
        poste(djorkaeff,md).
        poste(lizarezu,arg).
        poste(lizarezu,arm).
        affichageDetail(N):-not(joueur(N,S)),write("Joueur inconnu").
        affichageDetail(N):-joueur(N,S),poste(N,P),write("Joueur :", N,nl,"Nombre de selection :",
S,nl,"Poste :", P,nl).
        %listeJparPoste(P,L):-findAll(X,poste(X,P),L),affichageListeJ(L).
```

```
        listeJparPoste(P,L):-findAll(X,poste(X,P),L).
        affichageListeJ([T]):-write(" Joueur : ", T),nl.
        affichageListeJ([T|Q]):-write(" Joueur : ", T),nl,affichageListeJ(Q).
        listeMc(L):-findAll(X,poste(X,mc),L).
        listeMd(L):-findAll(X,poste(X,md),L).
        listeMg(L):-findAll(X,poste(X,mg),L).
        concatenerListe([],L,L).
        concatenerListe([X|Y],L,[X|R]):-concatenerListe(Y,L,R).
        contenir([T|Q],X):-T<>Q, contenir(Q,X).
        contenir([T|_],X):-T=X.
        test(L,X):-not(contenir(L,X)),write( X , " n'est pas un milieu"),nl.
        test(L,X):-contenir(L,X),write( X , " est un milieu"),nl.
        calculSomme([],0).
        calculSomme([T|Q],R):-joueur(T,S),calculSomme(Q,R2), R = R2 + S.
        tailleListe([],0).
        tailleListe([_|Y],L):-tailleListe(Y,L2), L = L2+1.
        calculMoy(X,Y,M):-M=X/Y.

goal
clearwindows,
write("Entrer le nom d'un joueur :"),nl,
realln(Nom),
affichageDetail(Nom),nl,
write("Entrer le poste :"),
readln(Poste),
listeJparPoste(Poste,L),
affichageListeJ(L),
listeMc(Lc),
listeMd(Ld),
listeMg(Lg),
ConcatenerListe(Ld,Lc,Ldc),
ConcatenerListe(Ldc,Lg,Lt),
affichageListeJ(Lt),
write("Entrer le nom d'un joueur :"),nl,
realln(Nom2),
test(Lt,Nom2),
findAll(X,joueur(X,_),L1),
calculSomme(L1,Somme),
tailleListe(L1,Taille),
calculMoy(Somme,Taille,Moyenne),
write("Moyenne : ", Moyenne).




domains
```

```
        couleur = symbol
        titre = symbol
        auteur = symbol
        theme = symbol
        prix = integer
        listeT = titre*
        listeA = auteur*

predicates
livre(couleur,titre,auteur)
%
caracteristique(couleur,theme,prix)
%
affichagelisteAvecPrix(auteur)
%
contenir(listA,auteur)
%
listeNoire()
%
listeRose()
%
listeVerte()
%
affichageListe(listeA,couleur)
%
testPlusieursCouleur(auteur)
%

affichageNbLivreParCouleur(auteur,couleur)
%

calculNbLivre(listeL,integer)
%

clauses
livre(noire,maitre_illusions,tart).
livre(verte,tour_sombre,king).
livre(rose,goeland,back).
livre(verte,thanatonautes,werber).
livre(verte,farenheit,bradbury).
caracteristique(noire,policier,15).
caracteristique(rose,roman,17).
caracteristique(verte,fiction,20).
affichagelisteAvecPrix(X):-not(livre(_,_,X)),write("Auteur non repertorie"),nl.
affichagelisteAvecPrix(X):-livre(C,T,X),caracteristique(C,_,P),write(" ",T," prix : ",P),nl.
contenir([],T).
contenir([V|Q],T):-V<>T,contenir(Q,T).
AffichageListe([],_).
AffichageListe([T|Q],_):-write(" ",T),AffichageListe(Q,_).
AffichageListe([T|Q],C):-write(C," Auteur :",T),AffichageListe(Q,_).
listeNoire():-findAll(A,livre(noire,_,A),L), AffichageListe(L,noire).
listeRose():-findAll(A,livre(rose,_,A),L), AffichageListe(L,rose).
```

```
listeVerte():-findAll(A,livre(verte,_,A),L), AffichageListe(L,verte).
TestPlusieursCouleur(A):-
findAll(A,livre(noire,_,A),L1),findAll(A,livre(rose,_,A),L2),findAll(A,livre(verte,_,A),L3),
contenir(L1,A)/\contenir(L2,A)\/contenir(L1,A)/\contenir(L3,A)\/contenir(L2,A)/\contenir(L3,A),w
rite("Auteur dans plusieur cat").
calculNbLivre([],0).
calculNbLivre([_|Q],R):-calculNbLivre([_|Q],R2), R = R2 + 1
AffichageNbLivreParCouleur(A,C):-findAll(T,livre(C,T,A),L),calculNbLivre(L,R),write("nb
livre :",R).
```

```
goals
clearwindow,
write("Nom de l'auteur :"),
readln(str1),
findall(str1,livre(_,X,str1),L),
write(L),
affichagelisteAvecPrix(L),
listeNoire(),
listeRose(),
listeVerte(),
TestPlusieursCouleur(str1).
```

```
domains
        ville = symbol
        distance = integer
        listeV = ville*

predicates
distanceTrain(ville,distance).
%DistanceTrain(bruxelles, 1000) signifie que la distance en train entre Toulouse et Bruxelles est de
1000
distanceAvion(ville,distance).
%DistanceAvion(rome, 1500) signifie que la distance en avion entre Toulouse et Rome est de 1500
affichageVille(listeV).
tailleListe(listeV,integer).
concatenerListe(listeV,listeV,listeV).
contient(listeV,ville).
chercherSiDesservie(listeV,listeV,ville).
```

```
clauses
distanceTrain(bruxelles,1000).
distanceTrain(barcelone,500).
distanceTrain(milan,1000).
```

```
distanceTrain(paris,800).
distanceAvion(rome,1500).
distanceAvion(londres,1000).
distanceAvion(tunis,2000).
affichageVille([]).
affichageVille([V|Q]):-write(V," "), affichageVille(Q).
tailleListe([],0).
tailleListe([_|Y],L):-tailleListe(Y,L2), L = L2+1.
concatenerListe([],L,L).
concatenerListe([X|Y],L,[X|R]):-concatenerListe(Y,L,R).
contient([V|_],X):-T=X.
contient([V|Q],X):-V<>X,contient(Q,X).
chercherSiDesservie(L1,_,X):-contient(L1,X),write(X," est desservie en train"), fail.
chercherSiDesservie(_,L2,X):-contient(L2,X),write(X," est desservie en avion").
chercherSiDesservie(L1,L2,X):-contient(L1,X),contient(L2,X),write(X," est desservie en avion et
en train").
chercherSiDesservie(L1,L2,X):-not(contient(L1,X)),not(contient(L2,X)),write(X," n'est pas
desservie").

goal
clearwindow,
write("Ville :"),
readln(str1),
findAll(X, distanceAvion(X,_),L),
findAll(X2, distanceTrain(X2,_),L2),
chercherSiDesservie(L,L2,str1).




%findAll(X, DistanceAvion(X,_),L)
%chercherSiDesservie([],[],X).
%chercherSiDesservie([V|Q],_,X):-X=V,write(X," est desservie en train").
%chercherSiDesservie([V|Q],_,X):-chercherSiDesservie(Q,_,X).
%chercherSiDesservie(_,[V|Q],X):-X=V,write(X," est desservie en avion").
%chercherSiDesservie(_,[V|Q],X):-chercherSiDesservie(_Q,X).
```