TP Prolog **2009/2010**
Master ILI & SIA

Exercice 1:
Une agence de voyages propose à ses clients des séjours de une ou deux semaines à Rome, Londres ou Tunis. Le catalogue de l'agence contient, pour chaque destination, le prix du transport (indépendant de la durée) et le prix d'une semaine de séjour qui varie selon la destination et le niveau de confort choisi : hôtel, chambre chez l'habitant ou camping.

1) Ecrire l'ensemble des assertions (règles) qui décrivent ce catalogue (les prix sont laisses à votre appréciation).
2) Exprimer la relation *voyage(V, D, H, S)* qui s'interprète par : le voyage dans la ville V pendant D semaines avec l'hébergement H coûte S euros.

```
voyage(V, D, H, S) :-
        transport(V, T),
        sejour(V, H, N),
        duree(D),
        S is N*D + T.
```

V : ville
D : durée, vaut 1 ou 2
H : hébergement
S : coût total
T : tarif du transport
N : prix du séjour pour 1 semaine

3) Compléter par *voyage-economique(V, D, H, S, SMAX)* qui exprime que le coût de ce voyage est inférieur à SMAX euros.

```
voyage-economique(V, D, H, S, Smax) :- voyage(V, D, H, S), S =< SMax.
```

Exercice 2 :
Nous sommes dans une agence matrimoniale qui possède un fichiers de candidats au mariage organisé par les assertions suivantes :

homme(N, T, C, A).
femme(N, T, C, A).

ou N est le nom d'un homme ou d'une femme, T sa taille (grande, moyenne ou petite), C la couleur de ses cheveux (blonds, bruns, roux, châtains), A son âge (jeune, mur ou vieux).

gout(N, M, L, S).

qui indique que la personne N aime le genre de musique M (classique, pop, jazz), le genre de littérature L (aventure, science-fiction, policier), et pratique le sport S (tennis, natation, jogging).

recherche(N, T, C, A).

qui exprime que la personne N recherche un partenaire de taille T, ayant des cheveux de couleur C et dont l'âge est A.

On considère que deux personnes X et Y de sexes différents sont assorties si X convient à Y et si Y convient à X. On dira que X convient à Y si d'une part X convient physiquement à Y (la taille, l'âge et la couleur des cheveux de X sont ceux que Y recherche) et si d'autre part les goûts de X et Y en matière de musique, littérature et sport sont identiques.

1)Donner un ensemble d'assertions représentant le fichier des candidats.

2)Ecrire les règles définissant *convient-physiquement(X, Y)*, puis les règles définissant ont-*memes-gouts(X, Y)*.

```
convient-physiquement(X, Y) :- homme(X, T, C, A),
                               femme(Y, T', C', A'),
                               recherche(X, T', C', A'),
                               recherche(Y, T, C, A).

convient-physiquement(X, Y) :- femme(X, T, C, A),
                               homme(Y, T', C', A'),
```

recherche(X, T', C', A'),
recherche(Y, T, C, A).

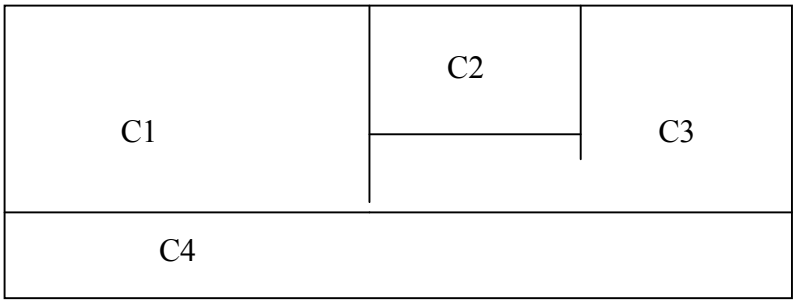
meme-gouts(X, Y) :- gouts(X, M, L, S), gouts(Y, M, L, S).

3)En déduire le programme qui détermine les couples assortis.

assortis(X, Y) :- convient-physiquement(X, Y), meme-gouts(X, Y).

Exercice 3 :

On se propose de définir un prédicat permettant de colorier la carte suivante :



Les règles sont les suivantes :

- On dispose de trois couleurs qui sont : vert, jaune et rouge ;
- Deux zones contiguës doivent avoir des couleurs différentes.

1) Ecrivez un prédicat *coloriage(C1, C2, C3, C4)* qui comportera deux parties. La première partie génère toutes les valeurs possibles de C1, C2, C3 et C4. La seconde vérifie si les colorations obtenues sont conformes à la carte par l'utilisation du prédicat *X \= Y* sur les couleurs des zones contiguës.

couleur(vert).
couleur(jaune).
couleur(rouge).

Coloriage(C1, C2, C3, C4) :- couleur(C1), couleur(C2), couleur(C3), couleur(C4),
C1\=C2, C1\=C3, C1\=C4,
C2\=C3, C3\=C4.

2) Reprenez ce prédicat, et modifiez le programme en déplaçant les tests de différence de couleurs le plus tôt possible dans l'écriture du prédicat, c'est-à-dire en vérifiant les différences de couleurs dès que celles-ci sont instanciées. Quelle en est la conséquence ?

Coloriage(C1, C2, C3, C4) :- couleur(C1), couleur(C2), C1\=C2,
couleur(C3), C1\=C3, C2\=C3,
couleur(C4), C1\=C4, C3\=C4.

Exercice 4 :

Une liste L qui contient un nombre pair d'éléments est donnée. Construire la liste M formée des éléments de rang pair de L.

1. En conservant leur ordre d'apparition dans L
2. En inversant cet ordre.

1) dans l'ordre

rangPair([], []).

```
rangPair([X, Y| L1], [X| L2]) :-
        rangPair(L1, L2).
```

2) dans l'ordre inverse

```
rangPair([], []).

rangPair([X, Y| L1], L2) :-
        rangPair(L1, L3),
        append(L3, [X], L2).
```

Exercice 5 :
Reprendre l'exercice précédent avec une liste comprenant un nombre impair d'éléments.

```
rangPair([], []).
rangPair([X], [X]).
rangPair([X, Y| L1], [X| L2]) :-
        rangPair(L1, L2).
```

Exercice 6 :
Deux listes U et V ont le même nombre d'éléments. Ecrire le programme *fusionner(U, V, L)* qui prend successivement un élément de U et de V pour construire L.
Exemple : U = [0, 1, 2, 3] et V = [4, 5, 6, 7], L = [0, 4, 1, 5, 2, 6, 3, 7]
En déduire un calcul direct des éléments de rang pair et impair d'une liste donnée L.

```
fusionner([], [], []).
fusionner([X| L1], [Y|L2], [X,Y| L3) :-
        fusionner(L1,L2, L3).
```

Exercice 7 :

Ecrire le prédicat element_de(X,L), vrai si X appartient à L.

```
element_de( X, [X | L]).
element_de ( X, [Y | L]) :- element_de ( X, L).
```

Modifier *element-de_bis* en ajoutant une règle et un argument, de façon à obtenir la réponse vraie si un élément donné X appartient à une liste donnée L.

```
element_de_bis(_,[],faux).
element_de_bis(X,[X|_],vrai).

element_de_bis(X,[U|L],R):-
        X\==U,
        element_de_bis(X,L,R).
```

Exercice 8 :
Donner les règles qui définissent *hors-de(X, L)* qui vérifient que l'objet X n'appartient pas à la liste L.

```
hors_de(_,[]).

hors_de(X,[U|L]):-
        X\==U,
        hors_de(X,L).

% ou encore en utilisant le not.
```

```
not(P):-P,!,fail.
not(_).

hors_de(X,L):-
        \+ element_de(X,L). % ou not
```

Exercice 9 :

Donner les règles qui définissent *differents(L)*, destiné à vérifier que la liste L ne contient pas de répétition.

```
differents([]).
differents ([X|L]) :-
        hors_de(X, L),
        differents(L).
```

Exercice 10 :

Donner les règles qui définissent *contenue(M, L),* lorsque tous les éléments de M sont dans L (dans un ordre quelconque).

```
contenue([], _).
contenue([X|M],L) :-
        element_de(X,L),
        contenue(M,L).
```

Exercice 11 :

Donner les règles qui définissent *debut-et-fin(U, L, V)*. La liste U est au début de la liste L, la liste V est la différence entre L et U.

Exemple : L = [0, 1, 2, 3, 4, 5], U = [0, 1] et V = [2, 3, 4, 5]

```
1)
debut_et_fin(U, L,V) :-
        append(U, V, L).
2)
debut_et_fin([], L,L).
debut_et_fin([X|L1], [X|L2], L3) :-
        debut_et_fin(L1, L2, L3).
```

Exercice 12 :

Dans cet exercice les listes représentent des ensembles. Il n'y a donc pas d'élément répété.

1. Donner les règles qui définissent *intersection(U, V, L),* où L est l'intersection de U et V.
2. Donner les règles qui définissent *union(U, V, L),* où L est l'union de U et V.

```
intersection([],_,[]).
intersection([X|L],Y,[X|Z]):-
        est_element_de(X,Y),
        intersection(L,Y,Z).
intersection([X|L],Y,Z):-
        hors_de(X,Y),
        intersection(L,Y,Z).

% union de deux listes

union([],L,L).
union([X|Y],L,[X|R]):-
        hors_de(X,L),
```

```
        union(Y,L,R).
union([X|Y],L,R):-
        est_element_de(X,L),
        union(Y,L,R).
```

Exercice 13 : Les dominos

On se donne une liste de dominos, chacun d'eux étant représenté par la liste [I, J] des deux chiffres qu'il porte. On cherche à les aligner, selon le procédé habituel :

- Un domino peut être ajouté à l'une des deux extrémités de la suite déjà construite.
- Les deux faces en contact doivent porter le même chiffre.

Partant d'un domino quelconque parmi ceux qui ont été donnés, construire une suite qui les contient tous.

```
domino(L,M):-
        permutation(L,M),
        aligne(M).

aligne([]).
aligne([D]).
aligne([ [_, Y], [Y, Z]|L):-
        aligne([[Y,Z]|L]).

permutation([],[]).
permutation([D|L],P):-
        permutation(L,L1),
        inserer(D,L1,P).

inserer([X,Y], L,[[X,Y]|L]).
inserer([X,Y], L, [[Y,X]|L]):-
        X\==Y.
inserer(D, [Y|L1], [Y|L2]):-
        inserer(D,L1,L2).
```

Exercice 14 :

Les sommets d'un graphe sont numérotés 0, 1, 2, 3, …, n. A chacun des arcs de ce graphe correspond une règle *fleche(I, J).* où I est l'origine et J l'extrémité de l'arc. Donner les règles qui définissent chemin(X, Y, L) où L est la liste représentant un chemin sans boucle qui conduit de X à Y.

```
fleche(1,2). fleche(1,3).
fleche(2,4). fleche(3,2).
fleche(4,3). fleche(4,5).
fleche(4,6). fleche(6,5).


chemin(X,Y,L):-
        chemin_s_boucle(X,Y,[X],L).
chemin_s_boucle(X,Y,L,[Y|L]):-
        fleche(X,Y).

chemin_s_boucle(X,Y,M,L):-
        fleche(X,Z),
        hors_de(Z,M),
        chemin_s_boucle(Z,Y,[Z|M],L).
```