Université d'Ottawa · University of Ottawa

Faculté de Génie – EECS

CSI2520 : PARADIGMES DE PROGRAMMATION

Tutorat 4

Exercice 1

Que fait ce prédicat:

```
premier([X|_],X).
```

Quel est le résultat de la requête ?

?- premier([2,4,6],2).
true.

Et celui-ci:

```
dernier([X],X).
dernier([_|L],X) :- dernier(L,X).
```

?- dernier([2,4,5],X).
X = 5 dernier([2,4,5],X)..

Et enfin celui-là

```
compte([],0).
compte([_|R],N) :- compte(R,N1), N is N1+1, N>0.
```

?- compte([2,4,5],X).
X = 3.

Exercice 2

Donnez la définition du prédicat occurrence (L,X,N) qui est vrai si N est le nombre de fois où X est présent dans la liste L.

```
occurence([],X,0).
occurence([X|T],X,Y):- occurence(T,X,Z), Y is Z+1.
occurence([X1|T],X,Z):- X1\=X, occurence(T,X,Z).
```

Exercice 3

Soit le programme suivant:

```
main :-
    open('c:/fruit.txt', read, Str),
    read_file(Str,Lines),
    close(Str),
    write(Lines), nl.

read_file(Stream,[]) :-
    at_end_of_stream(Stream).

read_file(Stream,[X|L]) :-
    \+ at_end_of_stream(Stream),
    read(Stream,X),
    read_file(Stream,L).
```

Quel est le résultat de la requête :

```
?- main.
```

Si le fichier fruit.txt contient les données suivantes:

```
pomme.
tomate.
orange.
celeri.
poire.
salade.
```

[pomme,tomate,orange,celeri,poire,salade]

Exercice 4

Comment peut-on définir le tri par insertion

tri_insertion(L,L1) : qui construit la liste triée L1 des éléments de L ?

```
sorting([A|B], Sorted) :- sorting(B, SortedTail), insert(A,
SortedTail, Sorted).
sorting([], []).

insert(A, [B|C], [B|D]) :- A @> B, !, insert(A, C, D).
insert(A, C, [A|C]).
```

Exercice 5

Comment peut-on définir le prédicat *permut* qui retourne toutes les permutations possibles d'une liste.

Voici un exemple d'utilisation :

```
/* Exemple :
?- permut([1,2,3],L).
L = [1,2,3] ;
L = [1,3,2] ;
L = [2,1,3] ;
L = [2,3,1] ;
L = [3,1,2] ;
L = [3,2,1] ;
no
*/
```

```
perm([],[]).
perm(L,[H|T]) :-
 append(V,[H|U],L),
 append(V,U,W), perm(W,T).
```

Exercice 6

Quatre personnages sont devant vous : un magicien, une magicienne, un sorcier, une sorcière. Chaque personnage a devant lui un sac rempli d'une ou plusieurs pièces de monnaie. Les pièces sont en bronze, en cuivre, en laiton ou en étain. Qui a le sac contenant le moins de pièces?
Sachant que :

1. Il n'y a pas deux sacs aux contenus identiques.
2. Dans un sac, il ne peut pas y avoir deux pièces de même métal.
3. Chaque sac contient, une, deux ou quatre pièces.
4. Le sorcier et le magicien ont chacun une pièce qu'aucun des trois autres possèdent.
5. Tous les sacs sans pièces de laiton comportent une pièce de bronze.
6. Tous les sacs sans pièces en étain n'ont pas non plus de pièces en bronze.
7. La magicienne a plus d'une pièce dans son sac.

Construire un programme Prolog utilisant une recherche en profondeur afin de trouver une solution à ce problème.

```
pieces([bronze,cuivre,laiton,etain]).
sac([X]):-pieces(P),member(X,P).
sac(S):-pieces(L),comb(2,L,S).
sac(S):-pieces(S).
valide(S):-sac(S),regle5(S),regle6(S).
solution([MM,SM,MF,SF]):-
valide(MM),valide(SM),valide(MF),valide(SF),MM\=SM,MM\=MF,MM\=SF,SM\=MF,SM\=SF,MF\=SF,regle4(
MM,SM,MF,SF),regle4(SM,MM,MF,SF),regle7(MF).
regle4(A,B,C,D):-member(X,A),\+member(X,B),\+member(X,C),\+member(X,D).
regle5(S):-member(laiton,S),!.
regle5(S):-member(bronze,S).
regle6(S):-member(etain,S),!.
regle6(S):- \+member(bronze,S).
regle7(S):-length(S,X),X>1.

comb(0,_,[]).
comb(N,[X|T],[X|Comb]):-N>0,N1 is N-1,comb(N1,T,Comb).
comb(N,[_|T],Comb):-N>0,comb(N,T,Comb).
```