Université de Limoges

PROLOG

TD 2

Exercice 1

Quelle est la réponse de Prolog pour les requêtes suivantes :

```
[a,b,c,d]  =  [a,[b,c,d]].
[a,b,c,d]  =  [a|[b,c,d]].
[a,b,c,d]  =  [a,b,[c,d]].
[a,b,c,d]  =  [a,b|[c,d]].
[a,b,c,d]  =  [a,b,c,[d]].
[a,b,c,d]  =  [a,b,c|[d]].
[a,b,c,d]  =  [a,b,c,d,[]].
[a,b,c,d]  =  [a,b,c,d|[]].
[]  =  _.
[]  =  [_].
[]  =  [_|[]].
```

Exercice 2

Quelles listes parmi les suivantes sont syntaxiquement correctes ? Donner la longueur pour celles-ci.

```
[1|[2,3,4]]
[1,2,3|[]]
[1|2,3,4]
[1|[2|[3|[4]]]]
[1,2,3,4|[]]
[[]|[]]
[[1,2]|4]
[[1,2],[3,4]|[5,6,7]]
```

Exercice 3

Décrire un prédicat `second(X,List)` qui nous dit si `X` est le deuxième élément de `List`.

Exercice 4

Écrire un prédicat `swap12(List1,List2)` qui teste si `List1` est identique à `List2` hormis les deux premiers éléments qui sont échangés. On pourra utiliser `Append/3` ou une méthode récursive.

Exercice 5

On dispose de la base de connaissance suivante :

```
tran(eins,one).
tran(zwei,two).
tran(drei,three).
tran(vier,four).
tran(fuenf,five).
tran(sechs,six).
tran(sieben,seven).
tran(acht,eight).
tran(neun,nine).
```

Écrire un prédicat `listtran(G,E)` qui traduit une liste de mots en allemand en les mots correspondants en anglais. Par exemple, `listtran([eins,neun,zwei],X).` doit renvoyer `X = [one,nine,two].`

Bien sûr, le programme devra fonctionner dans l'autre sens. Par exemple, si l'on demande `listtran(X,[one,seven,six,two]).` il devrait renvoyer `X = [eins,sieben,sechs,zwei].`

On pourra commencer par le cas de la liste vide puis procéder récursivement.

Exercice 6

Écrire un prédicat `twice(In,Out)` tel que l'argument de gauche est une liste, et tel que l'argument de droite doit être une liste redoublant les éléments de la première. Par exemple pour `twice([a,4,buggle],X).` Prolog devrait renvoyer `X = [a,a,4,4,buggle,buggle]).` De même, la requête `twice([1,2,1,1],X).` doit renvoyer `X = [1,1,2,2,1,1,1,1].`

Comme d'habitude, on pourra considérer le cas de la liste vide puis procéder récursivement.

Exercice 7

Écrire l'arbre de recherche pour les requêtes suivantes :

```
?- member(a,[c,b,a,y]).
?- member(x,[a,b,c]).
?- member(X,[a,b,c]).
```

Exercice 8

Écrire un prédicat `addone/2` dont le premier argument est une liste d'entiers et le second une liste d'entiers de même taille et dont les entiers sont obtenus en ajoutant 1 aux éléments de la première liste. Par exemple, la requête `addone([1,2,7,2],X).` doit renvoyer `X = [2,3,8,3].`

Exercice 9

On dit qu'une liste est doublée si elle est constituée de deux blocs consécutifs identiques d'éléments. Par exemple, `[a,b,c,a,b,c]` est doublée, de même que `[foo,gubble,foo,gubble]`. Par contre, `[foo,gubble,foo]` ne l'est pas. Écrire un prédicat `doubled(List)` qui réussit lorsque `List` est une liste doublée.

Exercice 10

Un palindrome est un mot ou une phrase qui s'écrit de la même manière de gauche à droite et de droite à gauche. Par exemple, 'rotator', 'eve', et 'nurses run' sont des palindromes. Écrire un prédicat `palindrome(List)` qui teste si `List` est un palindrome. Par exemple, Prolog répondra oui à `palindrome([r,o,t,a,t,o,r]).` et à `palindrome([n,u,r,s,e,s,r,u,n]).` mais non à `palindrome([n,o,t,h,i,s]).`

Exercice 11

Écrire un prédicat `toptail(OutList,InList)` qui dira oui si `InList` s'obtient à partir de `OutList` en supprimant son premier et son dernier élément. On demandera que `OutList` ait au moins deux éléments. Par exemple : `toptail([a],T).` renverra non. `toptail([a,b],T).` renverra `T=[]` et `toptail([a,b,c],T).` renverra `T=[b]`. On pourra utiliser `append/3`.

Exercice 12

Écrire un prédicat `last(List,X)` qui sera vrai si `List` est une liste de longueur au moins un et que dans ce cas, `X` en est son dernier élément. On proposera deux réponses : une utilisant une fonction `rev/2` qui renverse une liste et une autre utilisant une récursion. On testera sur quelques exemples si une méthode est plus efficace que l'autre.