Prolog - TP2
Récursion, Listes

Exercice 1 Quelques tests

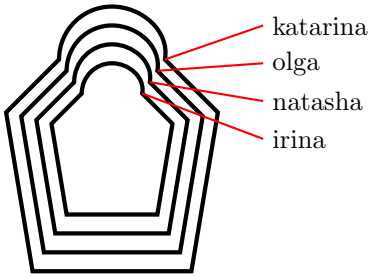
Sans lancer les requêtes, devinez quel est le résultat des requêtes suivantes :

1. `? [] = _.`
2. `? [_] = [].`
3. `? [_ | _] = [_].`
4. `? [_ | [_]] = [_].`

   Testez ensuite vos résultats avec Prolog.

Exercice 2 Poupées russes

On souhaite pouvoir représenter un jeu de poupées russes. Représentez la situation suivante :



Ecrivez le prédicat `dans/2`, vrai si une poupée est dans une autre peu importe son niveau d'imbrication. Vous pourrez vous appuyer sur un prédicat `directementDans/2`, vrai si une poupée est directement dans une autre.

Exercice 3 Dernier

Ecrivez le prédicat `dernier/2`, vrai lorsque le second argument est le dernier élément de la liste donnée en premier argument. Par exemple :

```
? dernier([a,b,c], c).
yes
```

Exercice 4 Inversion

1. Ecrivez le prédicat `inverse/2`, vrai lorsque la seconde liste est une inversion de la première liste. Par exemple :

```
? inverse([a,b,c,d], [d,c,b,a]).
yes
```

Vous pouvez éventuellement vous appuyer sur le prédicat `append/3`.

2. Ecrivez ensuite le prédicat `palindrome/1`, qui teste si une liste est un palindrome[1]. Par exemple :

```
? palindrome([n,o,n]).
yes
? palindrome([o,u,i]).
no
```

Testez `palindrome([o,u|Y])`..

3. Vous pouvez réécrire le prédicat `dernier/2` très facilement avec le prédicat `inverse/2`. Comment faire ?

Exercice 5 Plus longue

1. Ecrivez le prédicat `pluslongue/2`, vrai si la première liste est strictement plus longue que la seconde. Par exemple:

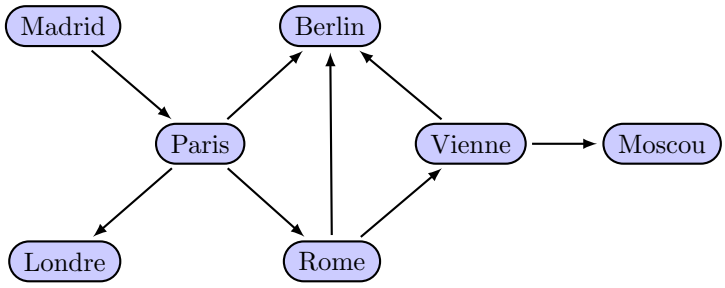
```
? pluslongue([a,b,c], [d,e]).
yes
```

2. Ecrivez le prédicat `lapluslongue/2`, tel que le premier argument soit une liste de listes, et le second argument la liste la plus longue strictement appartenant à cette liste. Par exemple :

```
? lapluslongue([[a,b], [c,d,e], []], X).
X = [c,d,e]
yes
```

Exercice 6 Voyage

On cherche à déterminer les trajets qui existent entre deux villes :



1. Ecrivez un prédicat `voyage/3`, dont le premier argument est une ville de départ, le second argument une ville de destination, et le troisième argument la liste des villes qu'il faut traverser pour se rendre de la ville de départ à la ville de destination. On doit avoir :

```
? voyage(madrid, vienne, [paris, rome, vienne]).
yes
```

[1] un palindrome est un mot qui se lit de la même façon de droite à gauche que de gauche à droite

Testez votre prédicat avec des variables, par exemple pour obtenir la liste des trajets possibles entre deux villes, ou l'ensemble des villes accessibles étant donné une ville de départ et un trajet.

2. Appuyez vous sur ce qui a été fait dans l'exercice précédent pour écrire le prédicat `lepluscourtvoyage/3`, qui renvoie le voyage le plus court entre deux villes. Vous pourrez utiliser le prédicat `findall/3` qui permet de rassembler dans une liste tous les résultats d'une requête. Par exemple, si on a :

```
p(a).
p(b).
p(c).
?findall(X, p(X), L).
L = [a,b,c].
```

Le premier argument désigne la variable correspondant aux éléments de la liste résultat, le second argument la condition que doit satisfaire la variable et le troisième argument désigne la liste résultat.