TP 2 - Prolog - Générer et Tester - S4

Un schéma souvent utilisé en programmation Prolog (et plus généralement en « résolution de problème ») est le suivant :

```
solve(Pb,Sol) :- guess(Pb, Sol), check(Pb,Sol)
```

Vision non déterministe : le prédicat `guess` « devine » une solution et `check` vérifie. On peut reformuler ce principe :

```
resoud (Pb,Sol) :- genere(Pb, Candidat), teste(Pb,Candidat), Sol =Candidat.
```

Vision déterministe : le prédicat `genere` « énumère » des » candidats » et `teste` ne garde que la ou les solutions effectivement correctes.

Exemples :
- Une phrase est représentée comme une suite de mots. On veut trouver le verbe :

```
verbe(Mot,Phrase) :-mem(Mot,Phrase), verbe(Mot).
```

- Trouver un élément commun à 2 listes.

```
commun(X, L1,L2) :- mem(X,L1), mem(X,L2).
```

Exercice 1 (**Racine carrée entière (Générer-tester brut)**)

1. Écrire un prédicat entier(N,I) tel que I est un entier supérieur ou égal à N.

   ```
   ?- entier(3,I).
   I=3; I=4; Etc.
   ```

2. En déduire un prédicat racine(N,R) calculant la racine carrée entière R d'un nombre entier N positif donné, c'est-à-dire un entier R tel que $R^2 \leq N < (R+1)^2$. Par exemple racine(5,R) donne R=2.
3. Calculer la racine carrée entière de 10, 20 et 1000.

Exercice 2 (**Carrés latins**)

On va appliquer cette technique pour générer des carrés latins.

[Wikipédia] Un carré latin est un tableau carré de n lignes et n colonnes remplies de n^2 éléments distincts dont chaque ligne et chaque colonne ne contient qu'un seul exemplaire.

Dans cet exercice, un carré latin est une matrice nxn qui contient les nombres de 1 à n^2 , disposés de telle façon que, pour chaque ligne et chaque colonne de la matrice, la somme des nombres de la ligne ou de la colonne en question donne le même résultat.

Par exemple, voici un carré latin de taille 3 :

8	3	4
1	5	9
6	7	2

On va représenter une telle matrice par une liste de n^2 éléments : les n premiers éléments de la liste représentent la première ligne de la matrice, les éléments du n + 1-ème au (2*n)-ième la deuxième ligne et ainsi de suite. Par exemple, la matrice ci-dessus est représentée par la liste [8, 3, 4, 1, 5, 9, 6, 7, 2]. On appelle cette liste une configuration.

Les questions 1 et 2 ci-dessous concernent la génération des configurations. Les questions 3 à 10 vont permettre de tester si une configuration est un carré latin. Dans la question 11, on rassemble tout pour définir un prédicat qui engendre les carrés latins de taille donnée.

1. Écrire un prédicat liste/2 tel que liste(N, L) est vrai si L est la liste des entiers de 1 à N.
2. En déduire un prédicat genere(N,L) qui est satisfait si et seulement si L est la liste des entiers compris entre 1 etle carré de N.
3. Définir un prédicat perm(L,G) qui exprime que G est une permutation d'une liste L donnée. On peut obtenir une permutation de L en prenant une permutation P de son reste et en insérant sa tête dans la permutation P..

```
?- perm([1,2,3], L).
L = [1, 2, 3] ;
L = [2, 1, 3] ;
L = [2, 3, 1] ;
L = [1, 3, 2] ;
L = [3, 1, 2] ;
L = [3, 2, 1] ;
no
```

4. Définir un prédicat nombre_latin(N,M) qui calcule le nombre magique de l'entier N (la division entière s'écrit // , infixe). Le nombre magique de N est $N(N^2+1)/2$.
5. Définir le prédicat somme_premiers(L, N, R) satisfait ssi R est la somme des des N premiers éléments de L.
6. Définir le prédicat elimine(L, N, M) qui est vrai ssi M est la liste L dont on a supprimé les N premiers éléments.i
7. Ecrire le prédicat somme_ligne(L, N, I, R) tel que R est la somme des éléments de la I-ème ligne de la configuration L d'un carré latin de taille N.
8. Ecrire le prédicat somme_colonne(L, N, I, R) tel que R est la somme des éléments de la I-ème ligne de la configuration L d'un carré latin de taille N.
9. Définir un prédicat lignes_ok(L,N,V) qui réussit si la somme des éléments sur chaque ligne de la matrice représentée par L est V.
10. Définir un prédicat colonnes_ok(L,N,V) qui réussit si la somme des éléments sur chaque colonne de la matrice représentée par L est V.
11. Définir le prédicat carre_latin(N,L) satisfait ssi L est la représentation d'un carré latin de taille N.
12. Définir une requête pour trouver tous les carrés magiques de taille 3. Combien y en-a-t-il ?
13. Pour les carrés magiques de taille 4, l'approche "générer et tester" montre ses limites. Pourquoi ?