Université Paris Diderot - M1 Informatique

TD de *Programmation logique et par contraintes* n° 4

"Générer et tester": les carrés magiques

Exercice 1 Un carré magique est une matrice $n \times n$ qui contient les nombres de 1 à n^2, disposés de façon telle que la somme des éléments de chaque ligne, de chaque colonne et des deux diagonales donne le même résultat. Ce résultat, qui ne depend que de n, est appelé *nombre magique* de n, et est égal à $\frac{n(n^2+1)}{2}$.

Par exemple, voici un carré magique de taille 3:

8	3	4
1	5	9
6	7	2

Le but de cet exercice est d'implementer un algorithme "générer et tester" pour engendrer les carrés magiques.

Une configuration est une liste de n^2 éléments, qui contient une permutation des entiers de 1 à n^2. Les n premiers éléments de la liste réprésentent 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 réprésentée par la liste `[8,3,4,1,5,9,6,7,2]`.

Les points 1 et 2 ci-dessous concernent la génération des configurations. Les points du 4 au 10 inclus vont permettre de tester si une configuration est un carré magique. Dans le point 11 on utilise le tout pour définir un prédicat qui engendre les carrés magiques de taille donnée.

1. Définir un prédicat `genere(+N,-L)` qui engendre une liste contenant les entiers de `N` à 1.
2. Définir un prédicat `perm(+L,-G)` qui engendre toutes les permutations d'une liste `L` donnée .
3. Définir un prédicat `nombre_magique(+N,-M)` qui calcule le nombre magique de l'entier `N` (la division entière s'écrit `//`, infixe).
4. Définir un prédicat `nth(+L,+N,-R)` qui renvoie le `N`-ième élément de `L`, et qui échoue si `N` est négatif ou nul, ou s'il majore la longueur de `L`. Par exemple, `nth([3,7,5],2,R)` donne `R=7`.
5. Définir un prédicat `accumuler(+L,+Init,+Step,+N,-R)` qui calcule la sommes des `N` entiers se trouvant aux positions `Init`, `Init+Step`, `Init + 2*Step+`, ...,`init + (N-1)*Step` de la liste `L`. Par exemple,

   `accumuler([1,2,3,4,5,6,7,8,9,10],1,2,4,R)`

   donne `R=16` (car 16=1+3+5+7).
6. utiliser `accumuler` pour définir des prédicats:
   - `somme_ligne(+L,+N,+I,-R)` qui calcule (en `R`) la sommme des éléments de la `I`-ème ligne de la configuration `L` d'un carré de taille `N`. (c.à.d. des éléments $(\mathtt{N}*(\mathtt{I}-1))+1,...,\mathtt{N}*\mathtt{I}$ de la liste `L`).

- `somme_colonne(+L,+N,+I,-R)`, l'analogue pour la somme des éléments de la `I`-ème colonne.

7. Définir un prédicat `test_lignes(+L,+N,+V)` qui réussit si la somme des éléments sur chaque ligne de la matrice réprésentée par `L` est `V`.
8. Définir un prédicat `test_colonnes(+L,+N,+V)`, analogue au précedent, pour les colonnes.
9. Définir un prédicat `somme_diagonale(+L,+N,+V)`, qui réussit si la somme des éléments de la diagonale est `V`.
10. Définir `somme_antidiagonale(+L,+N,+V)`, pour l'anti-diagonale.
11. Définir un prédicat `carre_magique(+N,-L)` qui engendre le carrés magiques de taille `N`.
12. Définir une requête pour trouver tous les carrés magiques de taille 3. Quel est leur nombre?
13. Pour les carrés magiques de taille 4, l'approche "générer et tester" montre ses limites. Pourquoi?