TD de *Programmation logique et par contraintes* n° 2

Récursion, I/O de base et listes.

Exercice 1 Pour chaque requête de la liste suivante, indiquer la réponse de l'interpreteur Prolog, puis vérifier:

1. `9 < 10.`
2. `X < 10.`
3. `X is 9+2, Y = X*2.`
4. `Y = X*2, X is 9+2.`
5. `X is Y+1, Y is 3.`
6. `X is 2, Y is X+1, X+Y<6`
7. `n(X,n(X,a,b),f(Y))=n(Y,n(a,a,b),f(Z)).`
8. `g(X,f(Y,X),A)=g(f(A),A,f(B)).`
9. `g(X,f(Y,X),A)=g(f(A),C,f(B)).`
10. `f(X,X)=f(g(a,Y),g(Z,b)).`
11. `f(g(X,a),Y)=f(Y,g(b,Z)).`

Exercice 2 Définir un prédicat `afficher/1` tel que, pour `n` entier naturel, `afficher(n)` affiche les entiers de `0` à `n` dans l'ordre (pour l'affichage, utiliser le prédicat prédéfini `write/1`).

Exercice 3 Définir un prédicat `boucle/0` qui affiche à l'ecran "choisir un entier", lit un entier, si l'entier donné est 0 termine, sinon repose la question est ainsi de suite. (Pour lire un entier, utiliser le prédicat prédéfini `read/1`; l'exécution de la requête `read(X)` a pour effet de donner la main à l'utilisateur, qui rentre un terme suivi d'un point. La variable X est alors unifiée avec ce terme).

Exercice 4 Si `x_1,...,x_n` sont des termes, `[x_1,...,x_n]` désigne la liste de ces termes, et si `x` est un terme et `l` une liste, `[x|l]` désigne la liste qui a `x` comme premier élément ("head") et `l` comme reste ("tail") (par exemple `[ 1|[2|[ ]]]=[1,2]`, `[ ]` étant la liste vide)

Donner le résultat de chacune des requêtes suivantes (travailler sur papier, puis lancer la requête pour vérifier).

1. `?- [a,[a]]=[H|T].`
2. `?- [[a,b],c]=[[H|T1]|T2].`
3. `?- [a,b,[c]]=[H1|[H2|[H3|T]]].`

- Définir le predicat `concat(X,Y,Z)`, vrai si `Z` est la concatenation de `X` et `Y`.
- Définir le prédicat `dernier(X,L)` qui réussit si `X` est le dernier élément de `L`. Donner deux versions de ce prédicat, avec et sans utilisation de `concat`.
- Définir deux prédicat (mutuellement récursifs) `longueurpaire(L)` est `longueurimpaire(L)` qui réussissentsi leur argument est une liste avec un nombre pair (impair) d'éléments (n'utilisez pas d'opérations arithmétiques).
- Définir le prédicat `reverse(L,L1)` pour inverser une liste. Par exemple `reverse([a,b,c],L)` donne `L = [c,b,a]`. (Pour ajouter un élément à la fin d'une liste on peut utiliser `concat`.). Est-ce que votre programme marche aussi avec `reverse(L,[a,b,c])` ?

Exercice 5

1. Écrire un prédicat `insert(Entier,ListeArgument,ListeResultat)` pour insérer un entier donné dans une liste d'entiers donnée, que l'on suppose triée, de telle façon que la liste résultat reste triée.

   Exemple d'utilisation:

   ```
     ?- insert(7,[3,5,13],R).
   R = [3,5,7,13] ?
   yes
   ```

2. Écrire un prédicat `tri(ListeArgument,ListeResultat)` pour trier une liste d'entiers. L'algorithme à utiliser est celui du "tri par insertion":

   - La liste vide est triée.
   - Pour trier la liste `[N|L]`, on commence par trier `L`, puis on insère `N` dans le résultat en utilisant `insert`.

   Exemple d'utilisation:

   ```
     ?- tri([5,3,13,1],R).
   R = [1,3,5,13] ?
   yes
   ```

3. rappel: le terme `[P,D|R]` désigne une liste dont le premier élément est `P`, le deuxième est `D` et le reste est la liste `R`.

   Écrire un prédicat `trie(ListeArgument)` qui réussit si l'argument est une liste triée, echoue sinon.

   Exemples d'utilisation:

   ```
     ?- trie([6,13,90]).
   yes
     ?- trie([4,2]).
   no
   ```

Exercice 6 On peut représenter les entiers naturels en *notation unaire* par les termes suivants:
`0,s(0),s(s(0)),s(s(s(0)))`,etc.

- Écrire un prédicat `transform(X,Y)` qui étant donné un entier en notation unaire `X` donne dans `Y` sa valeur. Par exemple

  `?- transform(s(s(s(0))),Y).`

  donne

  `Y=3.`

  Est-ce qu'on peut utiliser ce prédicat pour une requête de la forme `?- transform(X,3)` (qui devrait donner `X = s(s(s(0)))` ? Sinon définissez un prédicat pour cela.

- Sans utiliser `transform`, écrire un prédicat `somme(X,Y,Z)` qui prend deux entiers unaires et qui calcule leur somme. Par exemple

  `?- somme(s(s(0)),s(0),Z).`

  donne

  `Z = s(s(s(0)))`

  Est-ce que ça marche aussi pour `?- somme(X,Y,s(s(s(0)))).` ?

- Même chose pour le produit de deux entiers en utilisant `somme/3`.

- Même chose pour le calcul de n^m en utilisant `produit/3`.

Exercice 7 Indiquer ce que font les programmes suivants en général, et illustrer votre réponse sur le cas particulier proposé.

1. ```
   a(0,0).
   a(X,Y):- V is X-1, a(V,Z), Y is Z +  X.
   ```
   cas particulier: X=5.

2. ```
   b(0,0).
   b(X,Y):-V is X-1, b(V,Z),  Y is Z + 2*V + 1.
   ```
   cas particulier: X=9.

3. ```
   c(0,1).
   c(X,Y):- V is X-1, c(V,Z), Y is Z+Z.
   ```
   cas particulier: X=7.

4. ```
   d(X,0,1).
   d(X,Y,R):- Z is Y-1, d(X,Z,T), R is T*X.
   ```
   cas particulier: X=5,Y=3.