M2 Master Ingénierie Informatique 2014/2015
Programmation logique et par contraintes avancée Feuille de TP n° 1

Premier pas en Oz

Exercice 1
Écrire en Oz une procdure `Maxlist` à deux arguments telle que *f*`Maxlist L R`*g*, où L est une liste d'entiers non-négatifs, lie R à la valeur maximale de L (0 si la liste L est vide).

Exercice 2
Est-ce que le passage des paramètres en Oz est par valeur (c'est-à-dire, les arguments de l'appel d'une procédure sont évalués avant l'exécution du corps de la procédure) ou par nom (c'est-à-dire, les arguments de l'appel d'une procédure sont passés tels quels, et sont évalués seulement quand la procédure a vraiment besoin de la valeur d'un argument) ? Faites une expérience pour le tester.

Exercice 3
La procédure suivante pour la mise à jour d'une pair (clef, valeur) dans un arbre binaire recherche a été montrée en cours :

```
declare Insert in
proc {Insert Key Value TreeIn TreeOut}
   if TreeIn == nil then TreeOut = tree(Key Value nil nil)
   else
      local tree(K1 V1 T1 T2) = TreeIn in
         if Key == K1 then TreeOut = tree(Key Value T1 T2)
         elseif Key < K1 then
            local T in
               TreeOut = tree(K1 V1 T T2)
               {Insert Key Value T1 T}
            end
         else
            local T in
               TreeOut = tree(K1 V1 T1 T)
               {Insert Key Value T2 T}
            end
         end
      end
   end
end
```

Vous trouverez ce fichier également sur les PC de l'UFR à l'adresse `~treinen/plpc/tp1/insert.oz`.

1. Écrire une procédure qui, quand son permier argument est une liste de paires (clef,valeur), lie son deuxième argument à un arbre binaire de recherche qui contient toutes ces paires.

2. Écrire une procédure qui, quand son premier argument est un arbre binaire de recherche et son deuxième argument une clef, lie son troisème argument à la valeur stockée dans l'arbre pour cette clef.

3. Écrire une procédure qui, quand son premier argument est un arbre binaire de recherche et son deuxième argument une clef, lie son troisième argument à l'arbre sans la paire avec la clef donnée.

Exercice 4

Écrire la fonction de tri rapide d'une liste (quicksort) en Oz. La fonction prendra comme argument la liste à trier, et renvoie la liste triée.

Programmez l'algorithme de quicksort en utilisant les listes :

une liste de longueur 0 ou 1 est déjà triée

pour trier une liste d'au moins deux éléments on utilise le premier élément de liste comme *pivot*. La liste est partagée en deux sous-listes, une contenant les éléments strictement plus petits que le pivot, et l'autre les éléments qui sont égaux ou plus grands que le pivot. Puis on trie récursivement ces deux liste, et on recombine les résultats obtenus.

Indication : Il y a un constructeur de paires qui est notée #. Par exemple, `1#2` est la paire des valeurs 1 et 2. Vous pouvez facilement décomposer une paire p par une construction comme

```
local Left#Right = p in ... end
```