Premiers pas en Oz

Exercice 1
Écrire en Oz une procdure Maxlist à deux arguments telle que {Maxlist L R}, 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
La fonction suivante pour la mise à jour d'une pair (clef, valeur) dans un arbre binaire recherche a été montrée en cours :

```
% insertion of a new pair (key, value) into a binary search tree
declare
fun {FunInsert Key Value TreeIn}
   case TreeIn
   of nil then tree(Key Value nil nil)
   [] tree(K1 V1 T1 T2) then
         if Key == K1 then tree(Key Value T1 T2)
         elseif Key < K1 then
            tree(K1 V1 {FunInsert Key Value T1} T2)
         else
            tree(K1 V1 T1 {FunInsert Key Value T2})
         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 fonction qui, quand son premier argument est une liste de paires (clef,valeur), renvoye un arbre binaire de recherche qui contient toutes ces paires.

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

3. Écrire une fonction 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 3
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
```