TD de programmation fonctionnelle et logique

TD 6 : arbres

Arbres binaires

Un arbre binaire est soit une feuille (un entier), soit un nœud interne (composé d'un entier, la clef, et de deux sous-arbres). Nous définissons un arbre binaire par le type suivant :

```
#type arbre = Feuille of int | Noeud of (int * arbre  * arbre);;
 type arbre = | Feuille of int | Noeud of (int * arbre * arbre)
```

Fonctions élémentaires sur les arbres

1. Écrire une fonction `make_arbre` qui construit un arbre à partir d'un entier et de deux sous-arbres.
2. Écrire des fonctions `clé`, `fg` et `fd` qui extraient respectivement d'un nœud interne sa clé, son fils gauche et son fils droit.
3. Définissez une fonction `feuille` qui renvoie le booléen `true` si et seulement si son argument est une feuille.

Calcul sur les arbres

1. Définissez une fonction qui calcule la hauteur d'un arbre.
2. Construisez les fonctions `nb_feuilles` et `nb_noeuds` qui calculent respectivement le nombre de feuilles et de nœuds (nœuds internes et feuilles) d'un arbre.

Parcours dans les arbres

1. Écrivez une fonction `parcours_profondeur` qui énumère les clefs d'un arbre en profondeur d'abord.
2. Écrivez une fonction `appartient` qui renvoie `true` si et seulement si l'entier argument est contenu dans l'arbre étudié.
3. Écrivez une fonction `appartient_opt`, nouvelle version de `appartient` optimisée par l'usage d'exceptions.
4. Écrivez une fonction `parcours_largeur` qui énumère les clefs d'un arbre en largeur d'abord.