TD de programmation fonctionnelle et logique

TD 6 : exceptions

Listes

1. Écrivez une fonction qui recherche si un élément appartient à une liste.
2. Optimisez cette fonction par l'usage d'exceptions.
3. Écrivez une fonction booléenne récursive qui recherche si un élément appartient à une liste de listes.
4. Optimisez cette fonction par l'usage d'exceptions.
5. Écrivez, au moyen de fonctionnelles, une fonction booléenne qui recherche si un élément appartient à une liste de listes.
6. Optimisez cette fonction par l'usage d'exceptions.
7. Écrivez une fonction récursive qui recherche si un élément appartient à une liste de listes et qui renvoie la liste contenant l'élément.
8. Optimisez cette fonction par l'usage d'exceptions (on supposera ici que l'on traite des listes de listes d'entiers).
9. Écrivez, au moyen de fonctionnelles, une fonction qui recherche si un élément appartient à une liste de listes et qui renvoie la liste contenant l'élément.
10. Optimisez cette fonction par l'usage d'exceptions (on supposera ici que l'on traite des listes de listes d'entiers).

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)
```

1. Écrivez une fonction `appartient` qui renvoie `true` si et seulement si l'entier argument est contenu dans l'arbre étudié.
2. Écrivez une fonction `appartient_opt`, nouvelle version de `appartient` optimisée par l'usage d'exceptions.