Université Paris 7 - DEUG-MIAS Année 2002-2003

TP de *Logique et Circuits* n° 11

Induction sur les arbres

On représente les arbres binaires de type I étiquetés par des entiers par le type caml

```
type arbre = Av | Arb of int * arbre * arbre;;
```

Exercice 1

1. Ecrire une fonction `ajout_abr` telle que `ajout_abr n a` ajoute l'étiquette `n` à `a`. Lorsque `a` est un ABR, le résultat doit être encore un ABR.
2. Ecrire une fonction `rech_abr`, telle que `rech_abr n a` détermine, lorsque `a` est un ABR, si `n` apparaît dans `a`.

Exercice 2

1. Ecrire `list_of_arbre` telle que `list_of_arbre a` renvoie la liste des étiquettes de `a`, et telle que lorsque `a` est un ABR, cette liste soit triée.
2. Ecrire `arbre_of_list` telle que `arbre_of_list l` renvoie un arbre étiqueté par les éléments de `l`. Lorsque `l` est triée, le résultat doit être un ABR, équilibré au sens suivant :

   > Pour tout sous-arbre `Arb(n,g,d)`, `g` et `d` contiennent le même nombre d'étiquettes à une unité près.

   On écrira les fonctions intermédiaires suivantes :

   - `length` renvoyant le nombre d'éléments d'une liste
   - `diviser` telle que `diviser l n` renvoie un couple `(l1,l2)` où `l1` contient les `n` premiers éléments de `l`, `l2` les éléments suivants.
   - `separer` telle que `separer l` renvoie un couple `(l1,l2)` tel que `l1@l2 = l`, chaque liste contenant la moitié des éléments de `l`. Lorsque `l` est de longueur impaire, `l2` contiendra un élément de plus.

Exercice 3

1. Ecrire une fonction partielle `min_abr`, telle que `min_abr a` renvoie, lorsque `a` est un ABR non vide, le minimum de `a`.
2. Ecrire une fonction partielle `max_abr`, telle que `max_abr a` renvoie, lorsque `a` est un ABR non vide, le maximum de `a`.
3. A partir de ces deux fonctions, écrire une fonction `est_abr` déterminant si un arbre quelconque est un ABR.