TD numéro 1 : Manipulations d'arbres

Programmation avancée, ENSIIE

Semestre 2, 2012–13

Exercice 1 (Codage d'arbres en C)
Les arbres binaires non stricts sont des arbres pour lesquels chaque point de l'arbre à 0, 1 ou 2 fils. On a donc des feuilles, qui n'ont pas de fils, et des nœuds, qui ont 1 ou 2 fils. Pour les représenter, on considère des nœuds qui ont tous deux fils, ces fils pouvant éventuellement être l'arbre vide. On donne le type ML permettant des les représenter :

```
type 'a arbre_binaire =
  | Vide
  | Noeud of
    'a arbre_binaire * 'a * 'a arbre_binaire
```

Pour fixer les idées nous ne prendrons pas une représentation polymorphe en C mais nous fixerons `int` pour `'a`.

Question 1.1
En C, on représentera l'arbre vide par le pointeur `NULL`, et les arbres binaires par un pointeur vers une structure de nœud.

Définir le type `arbre_binaire` comme étant un pointeur vers une `struct donnee_arbre`.

Définir la `struct donnee_arbre` comme étant le produit d'un `int` et de deux `arbre_binaire`.

Question 1.2
Définir le test `est_feuille`, le test `est_noeud`, les fonctions `contenu_noeud`, `sous_arbre_gauche`, et `sous_arbre_droit` ainsi que la fonction `faire_noeud`.

Question 1.3 (Manipulation d'arbres)
On dira qu'un arbre réduit à une feuille est de hauteur 1. Combien de noeuds, respectivement de feuilles peut contenir un arbre de hauteur h ?

Écrire les fonctions `hauteur`, `nombre_de_feuilles` et `nombre_de_noeuds` et donner leur complexité.

Exercice 2
Un arbre t est dit équilibré si c'est une feuille ou si ses sous arbres gauches t_g et droits t_d sont équilibrés et si la différence de hauteur entre t_d et t_g est au plus 1. Donner la fonction `est_equilibre` ainsi que sa complexité.

Exercice 3 (Parcours d'arbres)
Les arbres peuvent servir à coder des expressions arithmétiques et nous souhaitons les imprimer de différentes manières. Le parcours préfixe d'un arbre consiste à traiter l'étiquette puis les sous arbres gauche et droit, le parcours infixe traite le sous arbre gauche puis l'étiquette puis le sous arbre droit alors que le parcours postfixe traite les sous arbres gauche et droit puis l'étiquette.

Écrire les fonctions `print_prefix`, `print_infix` et `print_postfix`. En C vous ferez l'affichage directement et en OCaml vous retournerez une chaîne de caractères.