TP n° 6 : Arbres

Pour sauver un arbre, mangez un castor — Henri Prades

On peut voir les arbres comme une généralisation de la structure de liste. À une tête (que l'on appelle plutôt *nœud*) on associe non pas une queue, mais deux (que l'on appelle sous-arbres) :

OCaml
```
type 'a arbre =
  | Vide
  | Noeud of 'a * 'a arbre * 'a arbre
```

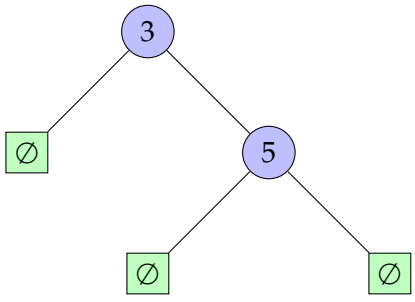

FIGURE 1 – Un exemple d'arbre avec deux nœuds. La racine d'étiquette 3 possède deux sous-arbres : son fils gauche qui est un sous-arbre vide et son fils droit qui est un sous-arbre de taille 1 (une feuille), dont l'étiquette est 5.

Un arbre non vide est de la forme `Noeud (etiquette, fils_gauche, fils_droit)`. Le nœud est appelée `racine` de l'arbre (l'analogue de la tête d'une liste) et on dit alors que `etiquette` est l'*étiquette* de ce nœud. Les sous-arbres `fils_gauche` et `fils_droits` sont respectivement le *fils gauche* et le *fils droit* de l'arbre (l'analogue de la queue de la liste, mais il y en a deux). On appelle *feuille* un nœud dont les deux fils sont des arbres vides.

Question 1

Définir en CAML l'arbre représenté sur la figure 1.

Question 2

Écrire une fonction `nb_noeuds : 'a arbre -> int` qui renvoie le nombre de nœuds d'un arbre (c'est l'analogue de la fonction `list_length`).

Question 3

Écrire une fonction `nb_feuilles : 'a arbre -> int` qui renvoie le nombre de feuilles d'un arbre.

Question 4

Écrire une fonction `somme_etiquettes : int arbre -> int` qui renvoie la somme des étiquettes des nœuds de l'arbre.

Question 5

Écrire une fonction `max_etiquettes : 'a arbre -> 'a` qui renvoie le maximum des étiquettes des nœuds de l'arbre.

Question 6

Écrire une fonction `somme_feuilles : int arbre -> int` qui renvoie la somme des étiquettes de toutes les feuilles.

Question 7

Écrire une fonction `parcours : int arbre -> int list` qui renvoie la liste des étiquettes d'un arbre, dans l'ordre que vous voulez, mais une et une seule fois chacune. On pourra utiliser l'opérateur `@`.

Question 8

Écrire une fonction `max_min_arbre : int arbre -> int * int` qui renvoie le minimum et le maximum des étiquettes d'un arbre. Chaque nœud ne devra être parcouru qu'une seule fois.

Question 9

Une branche est un chemin de la racine vers une feuille. Le poids d'une branche est la somme des étiquettes des nœuds qui la composent. Écrire une fonction `max_somme_branche : int arbre -> int` qui renvoie le poids de la branche de poids maximal.

Pour s'entraîner, on peut refaire tout le TP avec le type :

OCAML
```
type 'a arbre =
  | Feuille of 'a
  | NoeudInterne of 'a arbre * 'a * 'a arbre
```