TD - Arbres binaires

On travaille avec des arbres binaires munis d'étiquettes au niveau des feuilles et des nœuds internes. Une étiquette sera représentée par un élément de type `nat`.

1- Définir un type inductif polymorphe `arbre` permettant de décrire de tels arbres. Ce type sera paramétré par un type `nat`.

2- Construire le principe d'induction structurelle associé à la définition inductive de `arbre`.

3- Ecrire une fonction `nb_etiquettes` de type `arbre -> nat` qui compte le nombre d'étiquettes d'un arbre. Ecrire informellement les règles de calcul associés à cette définition.

4- Ecrire une fonction `mirror` qui échange récursivement les fils gauche et fils droit de chaque noeud.

Proposer une démonstration du fait que la fonction `mirror` est idempotente, c'est-à-dire que :

$$\forall a : \texttt{arbre},\ \texttt{mirror}\ (\texttt{mirror}\ a)\ =\ a.$$

5- Ecrire une fonction `map` permettant d'appliquer une même opération à chaque étiquette de l'arbre.

$$\texttt{map} : (\texttt{nat} \to \texttt{nat}) \to \texttt{arbre} \to \texttt{arbre}$$

Quelles sont ses propriétés au niveau du calcul ? En particulier que retourne la commande `Eval compute in (map f (noeud (feuille e1) e (noeud (feuille e2) (feuille e3)))).` en supposant que la fonction f n'est pas réductible quel que soit la valeur de son argument ?

6- Proposer une manière de démontrer en Coq que

$$\forall f\ g : \texttt{nat} \to \texttt{nat}, \forall a : \texttt{arbre}, \texttt{map}\ f\ (\texttt{map}\ g\ a)\ =\ \texttt{map}\ (\texttt{fun}\ (x : \texttt{int})\ \texttt{=>}\ f\ (g\ x))\ a.$$