TP n°3 bis

Arbres binaires – Bonus

1 Itérateur d'arbres

Exercice 1. Sans déclarer de fonctions auxiliaires et sans vous servir des fonctions prédéfinies, écrire :

```
forall_labels : ('a -> bool) -> 'a tree -> bool
```

telle que `forall_labels p a` renvoie `true` si et seulement si chaque étiquette `x` de `a` vérifie `(p x) = true`. A partir de cette seule fonction, écrire une fonction

```
is_uniform : 'a -> 'a tree -> bool
```

tel que `is_uniform v a` renvoit `true` si et seulement si chaque étiquette de `a` est égale à `v`. Cette fonction ne devra pas être déclarée comme récursive, mais doit déléguer entièrement la récurrence à la précédente.

Exercice 2. Toujours sans fonctions auxiliaires ou prédéfinies, écrire :

```
forall
```

renvoyant la somme des étiquettes d'un arbre étiqueté par des entiers.

Avec les mêmes contraintes, écrire :

```
map_tree : ('a -> 'b) -> 'a tree -> 'b tree
```

telle que `map_tree f a` renvoie l'arbre obtenu en remplaçant chaque étiquette `x` de `a` par `(f x)`.

2 Arbres binaires de recherche – exercices avancés

Exercice 5. Écrire la fonction suivante :

```
retire_abr : int -> int_tree -> int_tree
```

tel que, si `x` est un entier et `a` est un arbre, `retire_abr x a` est un arbre binaire de recherche qui ne contient pas `x`. Si `a` ne contient pas `x`, on retournera l'arbre inchangé.

Attention à préserver la propriété des ABRs. Vous testerez votre fonction sur0test'-3608()28(he)