université de BORDEAUX	**Année Universitaire 2017/2018, Session 1 d'automne** Licence Informatique & MI L2, UE 4TINA01U, Programmation fonctionnelle. Vendredi 24 novembre 2017. 16h30–18h Durée: 1h30. **Documents interdits.**	**Collège Sciences et Technologies**

L'évaluation attachera une grande importance à la clarté des justifications.
Seule la syntaxe vue en cours intégré et en TD est autorisée — Le barème est indicatif.

Exercice 1 — 4 points. 1) Écrire une fonction `split` de type

```
int -> int list -> (int list * int list)
```

qui, étant donnés un entier `n` et une liste `l`, renvoie un couple de listes `(l1,l2)` tel que :

- `l1` contient tous les éléments de la liste `l` qui sont inférieurs ou égaux à `n`.
- `l2` contient tous les éléments de la liste `l` qui sont strictement supérieurs à `n`.

L'ordre des éléments dans `l1` et `l2` n'a pas d'importance. En revanche, le nombre de fois où un élément apparaît dans `l` doit être le même que dans `l1` ou `l2`.

Par exemple, si `n` vaut `5` et `l` est la liste `[1;9;3;8;7;3;5]`, la fonction pourra retourner le couple `([1;3;3;5],[9;8;7])`, ou le couple `([5;3;3;1],[7;8;9])`. Votre fonction ne doit parcourir la liste `l` qu'une seule fois. Vous ne pouvez donc pas utiliser la fonction `List.filter`.

2) Votre fonction est-elle récursive terminale ? Justifiez votre réponse. Si ce n'est pas le cas, écrire une version récursive terminale de `split`.

Exercice 2 — 4 points. On considère le type suivant pour représenter les arbres binaires dont les feuilles contiennent des entiers.

```
type int_tree = Leaf of int | Bin of int_tree * int_tree
```

1) Écrire une fonction `substituer p t1 t2` qui remplace toutes les feuilles de `t1` dont l'étiquette satisfait le prédicat `p` par l'arbre `t2`. Par exemple, si `t1` vaut `Bin(Bin(Leaf 1, Leaf 2),Leaf 3)`, si `t2` est l'arbre `Bin(Leaf 0, Leaf 0)` et si le prédicat `p` est la fonction `fun x -> x = 3`, la fonction renverra l'arbre `Bin(Bin(Leaf 1, Leaf 2), Bin(Leaf 0, Leaf 0))`.

2) Quel est le type de votre fonction ?

Exercice 3 — 12 points. On considère le même type que dans l'exercice 2 :

```
type int_tree = Leaf of int | Bin of int_tree * int_tree
```

Un robot parcourt les nœuds de l'arbre (feuilles et nœuds binaires) en suivant les arêtes. Ses actions possibles sont descendre à gauche, descendre à droite et remonter une arête. On définit donc le type action suivant :

```
type action = Gauche | Droite | Remonter
```

Un parcours du robot est représenté dans le type `action list`.

1) Écrire une fonction `parcours_total` de type `int_tree -> action list` qui, étant donné un arbre en argument, renvoie une liste d'actions que doit effectuer le robot pour parcourir tout l'arbre en partant de la racine et en y revenant.

   Par exemple, pour l'arbre `Bin(Bin(Leaf 3, Bin(Leaf 91, Leaf 42)), Leaf 2)`, la fonction `parcours_total` pourra renvoyer la liste :

   ```
   [Gauche; Gauche; Remonter; Droite; Gauche; Remonter; Droite; Remonter;
    Remonter; Remonter; Droite; Remonter]
   ```

2) Donner un argument pour montrer que si un parcours part de la racine de l'arbre pour retourner à la racine de l'arbre, il a un nombre pair d'actions (on demande un argument mais pas une preuve formelle).

3) Le nombre d'opérateurs `::` effectués par votre fonction `parcours_total` sur un arbre à n feuilles est-il toujours proportionnel à n ? Si ce n'est pas le cas, proposer une version plus efficace de votre fonction qui utilise $O(n)$ opérateurs `::`.

4) On veut maintenant que le robot cherche dans son parcours une feuille étiquetée avec une valeur donnée. Écrire une fonction `parcours_recherche` de type :

   ```
   int_tree -> int -> action list
   ```

   qui à partir d'un arbre `t` et d'une valeur `x` à chercher, renvoie :

   - le parcours total de `t` dans le cas où `x` n'est pas l'étiquette d'une feuille de `t`,
   - le début de ce parcours qui s'arrête à la première feuille dont l'étiquette est `x`, si cette étiquette est dans `t`.

   Par exemple, si `t` vaut `Bin(Bin(Leaf 3, Bin(Leaf 91, Leaf 42)), Leaf 2)` et `x` vaut `42`, `parcours_recherche t x` renverra la liste :

   ```
   [Gauche; Gauche; Remonter; Droite; Gauche; Remonter; Droite]
   ```

5) Que doit-on changer dans le type et les fonctions précédentes pour traiter le cas d'arbres dont les feuilles sont d'un type quelconque ?