Université Paris Diderot — Programmation Fonctionnelle Avancée
M1 Informatique — Année 2013-2014

TP n°-

Zippers

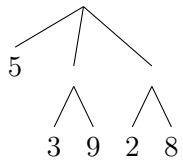
Rappel :

```
type 'a stack = 'a list
type 'a listzipper = 'a stack * 'a list

type 'a arbre = Feuille | Noeud of 'a * 'a arbre * 'a arbre

type marqueur = Gauche | Droite
type 'a block = marqueur * 'a * 'a arbre
type 'a path = 'a block list
type 'a arbrezipper = 'a path * 'a arbre
```

Exercice 1 [Manipulation pas à pas] Utilisez le toplevel afin de vous promener dans les différentes structures vues en cours (listes, arbres binaires, arbres n-aires). Pour les arbres n-aires, on pourra observer le résultat de `en_bas` puis `a_droite` sur l'exemple suivant :



Exercice 2 [Reconstruction de structures à partir de zipper] Il est tout à fait possible de retrouver à partir d'un zipper la structure de données originaire.

- en utilisant les fonctions du module List, écrivez une fonction `z_to_list : 'a listzipper -> 'a list` qui reconstruit la liste représentée par le zipper de liste `(pile, liste)` ;
- donnez une fonction `z_to_bintree : 'a arbrezipper -> 'a arbre` qui prend en argument un zipper d'arbre binaire et retourne l'arbre binaire représenté par celui-ci.

Essayez de trouver une façon générique de traiter cette problématique.

Exercice 3 [Parcours dans des arbres binaires] Ecrivez des fonctions récursives terminales définissant un parcours infixe et préfixe sur un arbre en utilisant des zippers. Vous pourrez en particulier utiliser la fonction `bas_a_gauche` pour descendre dans l'arbre et empiler ce qui reste à afficher.

Exercice 4 Donnez les types des structures de données suivantes, et dérivez-en automatiquement les blocs de pile pour leurs zippers :

- Arbre ternaire avec données contenues uniquement dans les noeuds
- Arbre ternaire avec données contenues uniquement dans les feuilles