TP numéro 4

1 Récursivité profonde

• Écrire une fonction qui calcule en profondeur le nombre d'occurrences d'un élément dans une liste.

• Écrire une fonction qui calcule la profondeur d'une liste.

```
(profondeur '(a (b c) d)) -> 2
(profondeur '(a (b) (c ((d e) f)))) -> 4
```

2 Arbres binaires

Commencez par implémenter les fonctions primitives sur les arbres vues en cours et en TD :
vide, vide?, valeur, fils-g, fils-d, cons-binaire, arbre=?, feuille ?

Pensez bien à tester vos fonctions sur un arbre où au moins un nœud a un seul fils.

• Écrire une fonction qui calcule le minimum d'un arbre binaire contenant des nombres.

• Écrire une fonction qui renvoie la liste des valeurs des feuilles d'un arbre.

• Écrire une fonction qui prend un arbre binaire et renvoie son miroir : pour tous les nœuds de l'arbre, le fils gauche devient le fils droit et le fils droit devient le fils gauche.

3 Listes de listes : une autre version du triangle de Pascal (cf. TD5)

• Écrire une fonction `som2` qui, étant donnée une liste de nombres, retourne la liste des sommes des éléments consécutifs de cette liste.

```
(som2 '(1 2 3 4)) -> (3 5 7)
```

• Utiliser la fonction `som2` pour écrire une autre version de la fonction qui construit une liste de listes correspondant aux n premières lignes du triangle de Pascal.

• Que pensez-vous de la complexité de cette version de la fonction par rapport à celle écrite en TD ?