EXERCICES SUR LES ARBRES BINAIRES

Les arbres binaires ont pour type

```
type arbre = V | S of int * arbre * arbre;;
```

Ce sont donc des arbres homogènes dont les étiquettes sont des entiers.

Sur le site du lycée (rubrique MP - informatique), vous trouverez un fichier contenant des exemples non triviaux d'arbres et de listes pour faire vos tests.

---

1) [Double parcours]

Montrez que si on connaît les deux listes des sommets d'un arbre binaire parcouru en profondeur préfixe et en profondeur infixe, alors on peut reconstituer l'arbre. Écrivez les fonctions nécessaires à la reconstitution de l'arbre.

Appliquez votre algorithme sur les listes du fichier précédent.

---

2) Linéarisation d'un arbre

Dans cet exercice, les arbres binaires sont supposés entiers : chaque nœud a zéro ou deux fils.

À tout arbre binaire, on associe la liste de ses éléments parcourus dans l'ordre suffixe, appelée liste suffixe de l'arbre.

a) Écrivez une fonction `liste_suffixe a` qui calcule la liste suffixe des éléments d'un arbre `a`. Donnez un exemple de deux arbres distincts qui ont même liste suffixe.

Lorsqu'on crée la liste suffixe des éléments, on perd la structure de l'arbre.

On peut faire mieux : au lieu de ranger dans la liste uniquement la valeur du sommet, on va aussi enregistrer si c'est une feuille ou un nœud (interne) grâce au type suivant

```
type contenu = F of int | N of int;;
```

a) Modifiez la fonction précédente pour qu'elle enregistre l'information supplémentaire dans la liste : la liste est appelée liste suffixe complète de l'arbre.

b) Écrivez la fonction réciproque, qui construit l'arbre associé à une liste suffixe complète. Si la liste n'est pas une liste suffixe complète, on donne un message d'erreur.

---

3) Construction d'un arbre aléatoire

Les exemples du fichier ont été créés aléatoirement selon le principe suivant :

— un arbre étant donné, on veut lui ajouter aléatoirement un objet : si l'arbre est vide, on n'a pas le choix ; sinon on tire un entier c au hasard entre 0 et 3 ; si $c = 0$, alors on ajoute l'objet aléatoirement dans le fils gauche ; si $c = 1$, on fait de même à droite : si $c = 2$, alors on met l'objet à la place de la racine et on place la racine aléatoirement dans le fils gauche ; si $c = 3$, on fait de même à droite ;

— on applique successivement le premier point : si $n = 0$, alors on construit l'arbre vide ; sinon on construit un arbre aléatoire à $n - 1$ éléments et on ajoute aléatoirement l'entier n.

Pour tirer un entier au hasard entre 0 et $k - 1$: `let c = random__int k`. Écrivez une fonction qui construit un arbre aléatoire contenant les entiers de 1 à n.

Plus difficile : construire un arbre aléatoire entier...