Interrogation 1 du Lundi 29 Septembre

Dans l'ensemble du devoir, toutes les fonctions seront :

- *écrites en langage OCaml,*
- *précédées d'une explication des variables utilisées,*
- *précédées d'une explication de l'algorithme.*

Exercice 1 (Du cours)

1. (a) Prouver par induction structurelle que tout arbre binaire A à n sommets et de hauteur h vérifie
$$h + 1 \leqslant n \leqslant 2^{h+1} - 1.$$
   (b) Démontrer que, si A est un arbre binaire parfait à n sommets, alors sa hauteur vaut $\lfloor \log_2(n) \rfloor$.
2. (a) Donner une définition des arbres binaires de recherche.
   (b) Soit A un arbre binaire de recherche à étiquettes dans un ensemble totalement ordonné $(\Sigma, \leqslant)$.
   Démontrer que, lors d'un parcours en profondeur en mode infixe de A, les étiquettes sont parcourues par ordre croissant.

Exercice 2 (Arbres peignes)

On suppose défini le type arbre de la manière suivante :

```
type arbre =
  | Feuille of int
  | Noeud of arbre * arbre
;;
```

On dit qu'un arbre est un **peigne** si tous les noeuds à l'exception éventuelle de la racine ont au moins une feuille pour fils.

On dit qu'un peigne est **strict** si sa racine a au moins une feuille pour fils, ou s'il est réduit à une feuille.

On dit qu'un peigne est **rangé** si le fils droit d'un noeud est toujours une feuille.

Un arbre réduit à une feuille est un peigne rangé.

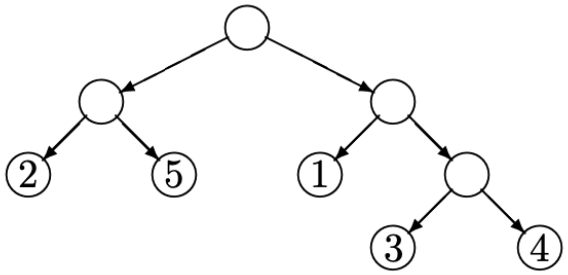


Figure 1 : un peigne à cinq feuilles

1. Représenter un peigne rangé à 5 feuilles.

2. La **hauteur** d'un arbre est le nombre d'arêtes maximal que l'on rencontre pour aller de la racine à une feuille (la hauteur d'une feuille seule est 0). Quelle est la hauteur d'un peigne rangé à n feuilles ? On justifiera soigneusement la réponse.
3. Écrire une fonction `est_range : arbre -> bool` qui renvoie `true` si l'arbre donné en argument est un peigne rangé.
4. Écrire une fonction `est_peigne_strict : arbre -> bool` qui renvoie `true` si l'arbre donné en argument est un peigne strict.

   En déduire une fonction `est_peigne : arbre -> bool` qui renvoie `true` si l'arbre donné en argument est un peigne.
5. On souhaite ranger un peigne donné. Supposons que le fils droit N de sa racine ne soit pas une feuille.

   Notons A_1 le sous-arbre gauche de la racine, f l'une des feuilles du noeud N et A_2 l'autre sous-arbre du noeud N. On va utiliser l'opération de **rotation** qui construit un nouveau peigne où
   - le fils droit de la racine est le sous-arbre A_2 ;
   - le fils gauche de la racine est un noeud de sous-arbre gauche A_1 et de sous-arbre droit la feuille f.

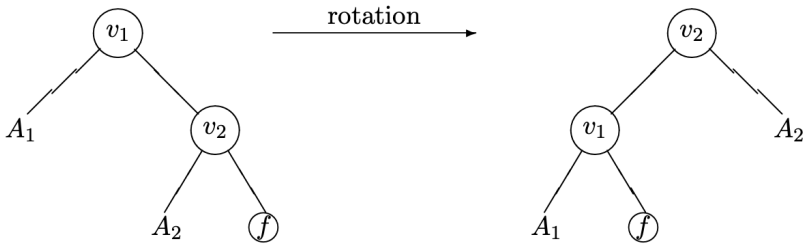


Figure 2 : une rotation

(a) Donner le résultat d'une rotation sur l'arbre de la figure 1.

(b) Écrire une fonction `rotation : arbre → arbre` qui effectue l'opération décrite ci-dessus. La fonction renverra l'arbre initial si une rotation n'est pas possible.

(c) Écrire une fonction `rangement : arbre → arbre` qui range un peigne donné en argument, c'est-à-dire qu'elle renvoie un peigne rangé ayant les mêmes feuilles que celui donné en argument. La fonction renverra l'arbre initial si celui-ci n'est pas un peigne.

Exercice 3 (Arbres rouge-noir)

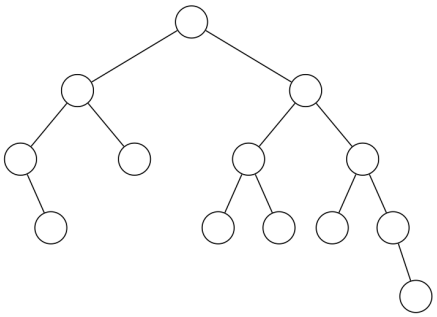
On appelle **arbre bicolore** un arbre binaire comportant un champ supplémentaire par nœud : sa couleur, qui peut être rouge ou noire. Ceci nous conduit à définir les types :

```
type couleur = Rouge | Noir ;;

type bicolore = Vide | N of bicolore * couleur * bicolore ;;
```

Un arbre bicolore est un **arbre rouge-noir** s'il vérifie les conditions suivantes :
- la racine est noire ;
- chaque nœud rouge n'a pas de fils rouge ;
- pour chaque sommet, tous les chemins le reliant à des feuilles (`Vide`) contiennent le même nombre de nœuds noirs.

1. Montrer que l'arbre ci-dessous peut être muni d'une coloration rouge-noir.



2. Donner un exemple d'arbre qui ne puisse pas être muni d'une coloration rouge-noir.
3. Étant donné un arbre rouge-noir A, on appelle **hauteur noire** de A et on note $b(A)$ le nombre de noeuds noirs que contient chacun des chemins d'un `Vide` à la racine (indépendant du choix de la feuille par définition).
   (a) Montrer que : $b(A) \leqslant h(A) + 1 \leqslant 2b(A)$.
   (b) Montrer par induction structurelle que : $|A| \geqslant 2^{b(A)} - 1$.
   (c) En déduire que les arbres rouge-noir sont équilibrés.
4. (a) Écrire en OCaml une fonction `couleur_fils : bicolore -> bool` qui, étant donné un arbre bicolore, vérifie si aucun nœud rouge n'a de fils rouge.
   (b) Écrire en OCaml une fonction `hauteur_noire : bicolore -> bool` qui, étant donné un arbre bicolore, vérifie que le nombre de nœuds noirs entre une feuille (`Vide`) et la racine est constant.
   (c) Écrire en OCaml une fonction `rouge_noir : bicolore -> bool` qui, étant donné un arbre bicolore, vérifie s'il est un arbre rouge-noir.