Algorithmique des structures de données arborescentes
Feuille d'exercices 1

1.1 Révisions : Listes

Exercice 1 Écrire une fonction `mirror` de type `'a list -> 'a list`, qui retourne la liste des éléments dans l'ordre inverse de celui de son argument.

Exemple : `mirror [1; 2; 3; 4; 5]` doit renvoyer la liste `[5; 4; 3; 2; 1]`.

Note. Cette fonction existe déjà dans la bibliothèque standard OCaml, elle s'appelle `List.rev`.

Exercice 2 Écrire en OCaml une fonction `append` de type `'a list -> 'a list -> 'a list` qui concatène deux listes (sans utiliser l'opérateur de concaténation `@` de OCaml).

Exemple : `append [1; 2; 3] [4; 5]` s'évalue en `[1; 2; 3; 4; 5]`.

Note. Cette fonction existe dans la bibliothèque standard OCaml, elle peut être utilisée sous le nom `List.append`. Voir https ://caml.inria.fr/pub/docs/manual-ocaml/libref/List.html.

Exercice 3 Écrire une fonction `apply` de type `('a -> 'b) -> 'a list -> 'b list`, telle que `apply f l` retourne la liste obtenue en appliquant la fonction `f` à tous les éléments de `l`.

Exemple : `apply (fun x -> 2*x) [1; 2; 3; 4; 5]` doit renvoyer la liste `[2; 4; 6; 8; 10]`.

Note. Cette fonction existe déjà dans la bibliothèque standard OCaml, elle s'appelle `List.map`.

Exercice 4

- Écrire une fonction `sum` de type `'a list -> int`, qui retourne la somme des éléments de la liste passée en argument.
  Exemple : `sum [1; 2; 3; 4; 5]` doit renvoyer 15.
- De manière similaire, écrire une fonction `prod` qui calcule le produit des éléments de la liste passée en argument.
- Les deux fonctions précédentes suivent un modèle générique. Écrire une fonction `replier` de type `('a -> 'b -> 'a) -> 'a -> 'b list -> 'a` telle que `replier f a [b1; ...; bn]` renvoie `f (... (f (f a b1) b2) ...) bn`.
- Réécrire `sum` et `prod` en utilisant `replier`.

Révisions : Types

Exercice 5

1. Écrire un type `couleur` ayant trois valeurs : `Blanc`, `Rouge` et `Rose`.
2. On définit le type `region` comme suit :

   ```
   type region = Medoc | Graves | Alsace | Beaujolais | Touraine | Bourgogne
   ```

   En utilisant les types `couleur` et `region`, définir un type `vin` permettant de représenter un vin par
   - sa région,
   - sa couleur, **et**
   - son millésime (c'est-à-dire, une année, de type `int`).
3. Écrire une fonction `bordeaux` qui prend en paramètre une liste de vins et renvoie la sous-liste des vins de Bordeaux (c'est-à-dire produits dans le Médoc ou dans les Graves) qu'elle contient.
4. Écrire une fonction `millesimes` qui prend en paramètres une liste de vins `l` et un prédicat `p` de type `vin -> bool`, et qui renvoie la liste des millésimes des vins de `l` qui satisfont `p`. Donner un exemple d'appel de votre fonction, ainsi que le type et la valeur de son retour.

Exercice 6

1. Définir un type `carburant` ayant trois constructeurs `Diesel`, `Essence` ou `Electrique`.
2. Un *véhicule* est caractérisé par son carburant et son nombre de roues. Définir un type `vehicule` répondant à ces critères.

3. Lors des pics de pollution, les véhicules diesel à 4 roues au moins sont interdits. Écrire une fonction `peut_rouler : vehicule -> bool` qui teste si un véhicule est autorisé.

4. Pour rouler 100km, un véhicule électrique consomme environ 10kWh, un véhicule diesel consomme environ 6L de carburant, et un véhicule essence consomme environ 8L. Sachant qu'1kWh coûte 0.25 et qu'un litre de carburant coûte 1.5, écrire une fonction `consommation : vehicule -> int -> float` telle que `consommation v n` renvoie le coût d'utilisation du véhicule `v` sur `n` kilomètres.

5. Ajouter un constructeur au type `carburant` de façon à prendre en compte les véhicules hybrides (pouvant fonctionner avec deux types de carburants), et donner un exemple de véhicule hybride.

1.2 Rappels sur les arbres binaires

Un ***arbre binaire*** `T` est

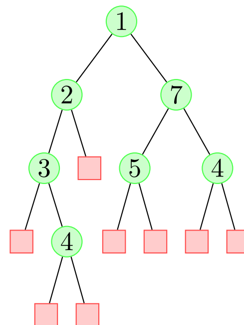
- soit l'arbre vide,
- soit constitué :
  - d'un nœud `r` pouvant porter une valeur, appelé la *racine* de l'arbre,
  - d'un arbre `G`,
  - d'un arbre `D`.

  On voit que la définition est récursive, puisqu'un arbre non vide est constitué de deux sous-arbres (et de sa racine).
- L'arbre `G` s'appelle le ***sous-arbre gauche*** de `T`. Si l'arbre `G` n'est pas vide, sa racine est appelée ***fils gauche*** du nœud racine `r` de `T`.
- De même, l'arbre `D` s'appelle le ***sous-arbre droit*** de `T`. Si l'arbre `D` n'est pas vide, sa racine est appelée ***fils droit*** du nœud racine `r` de `T`.

En OCaml, on représente un arbre binaire avec étiquettes de type `'a` par le type suivant :

```
type 'a btree =
| Empty
| Node of 'a * 'a btree * 'a btree
```

Une ***feuille*** est un nœud qui n'a pas de fils. Un ***nœud interne*** est un nœud qui n'est pas une feuille. Par exemple, l'arbre suivant comporte 7 nœuds dont 3 feuilles et 4 nœuds internes. Notez que deux feuilles portent la même étiquette : 4. Le fils gauche de la racine est le nœud étiqueté 2. Son fils droit est le nœud étiqueté 7.



On dit qu'un arbre est ***complet*** si tout nœud interne a exactement deux fils. Une ***branche*** d'un arbre t est une suite de nœuds allant de la racine à une feuille (sans "remonter"). Formellement, c'est une suite de nœuds $n_0, n_1, \ldots, n_k$ où n_0 est la racine de t, n_k est une feuille de t, et pour chaque i, n_{i+1} est un fils (droit ou gauche) de n_i. La ***hauteur*** de t est le nombre de nœuds de sa plus longue branche, moins 1. La hauteur de l'arbre vide est par convention -1.

1.3 Propriétés importantes des arbres binaires

Exercice 7 1. Trouver des relations satisfaites par les fonctions de hauteur, taille, nombre de nœuds internes et nombre de feuilles, entre un arbre binaire et ses deux sous-arbres gauche et droit.

2. En déduire des fonctions récursives OCaml pour calculer la hauteur, la taille, le nombre de nœuds internes et de feuilles d'un arbre binaire donné en argument.

Exercice 8 On note $h(t)$ la hauteur et $n(t)$ le nombre de nœuds d'un arbre binaire t. Montrer :

a) Qu'on a $h(t)+1 \leqslant n(t) \leqslant 2^{h(t)+1}-1$.

b) Que l'arité de tous les nœuds internes de t est 1 si et seulement si $h(t)+1 = n(t)$.

c) Que l'arbre t est parfait si et seulement si $n(t) = 2^{h(t)+1}-1$.

Exercice 9 Pour un arbre binaire plein t non vide avec $n(t)$ nœuds, $l(t)$ feuilles et $i(t)$ nœuds internes, montrer les deux relations suivantes. Sont-elles vraies pour tous les arbres binaires ?

$$l(t) = 1 + i(t) \quad \text{et} \quad n(t) = 2l(t) - 1 = 2i(t) + 1.$$

Exercice 10 Dans un arbre binaire, quel est le nombre maximal de nœuds à profondeur d ? Justifier.

1.4 Notation O()

Exercice 11 Exprimer toutes les relations de la forme $f = O(g)$ pour f et g parmi les fonctions suivantes : $f_1(n) = 42n^7+12n^5-12345$, $f_2(n) = (\log(n))^{42}$, $f_3(n) = n\log_2(n)$, $f_4(n) = n\log_{10}(n)$, $f_5(n) = n^7$, $f_6(n) = n^{0.001}$, $f_7(n) = 1.001^n$ et $f_8(n) = 2^n$.