TP 3 ter

1 Tri fusion

1. Implémenter le tri fusion sur des listes. On demande à ce que la fonction puisse être utilisé pour différents critère de comparaisons, et sur des listes d'éléments de différents types.
2. Créer un type adapté à représenter un ensemble de figure géométrique : des cercles de rayons donnés, des carrées de cotés donnés, des rectangles de cotés donnés, de triangles rectangles dont la longueurs des cotés sont données. Écrire une fonction qui calcule l'air d'une figure géométrique.
3. Créer une liste de figures et triez-les par ordre croissant d'aire.

2 Mots bien parenthésés

Un mot bien parenthésé est une mot sur un alphabet à 3 lettre (en général `(, ) et -`) tel que chaque préfixe contienne plus ou autant de `(` que de `)`, et que le mot complet contienne autant de `(` que de `)`. Écrire une fonction récursive terminale permettant de tester si un mot sur les trois lettres données (représenté par une liste) est bien parenthésé.

3 Arbres

Un arbre binaire est soit une feuille, soit un couple (fg, fd), où fg et fd sont des arbres binaires.

1. Définissez un type `arbre_binaire` en OCaml.
2. Un processus de Galton-Watson de paramètre p permet de générer aléatoire de arbres. Pour cela, on tire au hasard si l'arbre est réduit à une feuille (avec probabilité $1 - p$), ou s'il a deux fils (avec probabilité p). Dans le cas où l'arbre possède 2 fils, on appelle récursivement le même processus. Coder un générateur aléatoire d'arbre basé sur ce principe.
   Attention : Le processus ne termine pas toujours. En particulier, si $p > 1/2$, l'espérance de taille de l'arbre obtenu est infinie...
3. Un arbre binaire complet est un arbre ayant toutes ses feuilles à même profondeur. Codez une fonction permettant de générer un arbre complet de profondeur n (l'arbre réduit à une feuille a une profondeur de 0).
4. Écrire une fonction permettant de générer une liste contenant tous les arbres de taille n.

4 Listes d'associations

Une *liste d'association* est une représentation d'association de clés à valeur. Par exemple, on pourrait considérer un petit annuaire téléphonique en deux temps

1. associer un nom à une liste de coordonnées
2. associer un "type" de coordonnées à la valeur associée

On pourrait avoir un annuaire de la forme suivante

```
[
  ("Sophie",
    [("phone", "0656453456") ;
     ("adress", "5 av Gambetta, Paris")]) ;
  ("Mark",
    [("email", "mark@mark.net) ;
     ("phone", "0785432312") ;
     ("fax", "0134675698")])
 ]
```

1. Écrire les fonctions suivantes

(a) `val assoc : 'a -> ('a * 'b) list -> 'b`
`assoc a l` retourne la valeur associée à la clé `x` dans la liste de paires `l`. Elle déclenche l'exception `Not_found` si elle n'existe pas ;

(b) `val remove_assoc : 'a -> ('a * 'b) list -> ('a * 'b) list`
`remove_assoc a l` retourne la liste de paires `l` sans la première paire de clé `a` si elle existe.

2. Donner une fonction `annuaire` qui à un nom `nom` et une liste d'association `l` associe le numéro de téléphone associé.
3. Il est aussi possible de manipuler directement les valeurs des listes d'associations.
   Donner une fonction `annuaire_inverse` qui à un numéro de téléphone `num` et une liste d'association `l` associe le nom de la personne.
4. Proposer une variante utilisant des types dédiés à ce problème.

5 Exponentiation rapide

1. Écrire une fonction `exp : float -> int -> float` qui à un décimal x et un entier n associe le décimal x^n. Quelle est la complexité de votre algorithme en l'entier n ?
2. Écrire la fonction `fast_exp` en utilisant le fait que

$$\begin{aligned} x^{2n} &= (x^n)^2 \\ x^{2n+1} &= x \times (x^n)^2 \end{aligned}$$

Désormais, quelle est la complexité de cette fonction en l'entier n ?

6 Méthode de Newton

En analyse numérique, les problèmes d'approximation font partie des grandes questions ouvertes des mathématiques. Nous nous intéressons dans cet exercice à approximer les racines de fonctions, c'est-à-dire les réels x_0 tels que $f(x_0) = 0$, en utilisant la méthode de Newton-Raphson. Cette méthode ne donne pas une valeur exacte mais une fine approximation, il s'agit alors d'itérer la méthode jusqu'à l'obtention d'un certain x_0 à une précision ε près.

1. Écrire une fonction `derive` qui à une fonction f, un décimal h suffisamment petit et un décimal x calcule l'approximation de la dérivée de f en x donnée par

$$f'(x) = \frac{f(x+h) - f(x)}{h}$$

2. Écrire une fonction `assez_precis` qui à deux décimaux y et ε décide si $-\varepsilon < y < +\varepsilon$;
3. Écrire une fonction `iterer_newton` qui à une fonction f, un décimal z_n, une précision ε, calcule le décimal suivant

$$z_{n+1} = z_n - \frac{f(z_n)}{f'(z_n)}$$

4. Donner une fonction `newton` qui prend en paramètres une fonction f, une précision ε et un décimal z_0 et retourne une approximation de racine de f avec la précision ε.
5. Tester votre solution avec la fonction f définie sur l'intervalle $[-1; +\infty]$ par la valeur $f(x) = \sqrt{1+x}$. Vérifiez la correction de la valeur obtenue en résolvant [1] l'équation $f(x) = x$.

7 Générateur de spam [Treinen, 2010]

Le but de cet exercice est générer un texte pseudo-aléatoire à partir d'un texte de référence à l'aide d'une table de succession, sous la forme d'une liste d'association.

1. Il s'agit d'un polynôme du second degré...

Table de succession Il s'agit tout d'abord de construire la table de succession d'un texte de référence. Prenons le texte suivant dû à Aristote :

« Le faux et le vrai ne sont pas dans les choses, comme si le bien était le vrai et le mal, en lui-même le faux, mais dans la pensée, et en ce qui regarde les natures simples et les essences, le vrai et le faux n'existent pas même dans la pensée. [2] *»*

En faisant abstraction des majuscules et en considérant l'espace comme l'unique séparateur de mots, le mot "le" admet huit occurences dans le texte et peut être suivi des mots "faux" et "vrai". Tandis que le mot "pensée." apparaît deux fois et n'admet aucun successeur. On peut alors construire la table de succession suivante :

```
[("le",   ["faux"; "vrai"; "faux,"; "mal,";
           "vrai"; "bien"; "vrai"; "faux"]);
 ("faux", ["n'existent"; "et"]);
 ("et",   ["le"; "les"; "en"; "le"; "le"]); ...]
```

Génération de texte La génération d'un texte aléatoire a partir d'une table de succession se fait par la méthode suivante : le premier mot du texte généré est choisi au hasard dans la table. A chaque étape, on prend la liste associé au mot courant dans la table, et le mot suivant est choisi au hasard dans cette liste. Si le mot courant est le dernier, le mot suivant est à nouveau choisi au hasard dans la table.

On peut alors produire le texte suivant :

« pas même dans la pensée, et le bien était le mal, en ce qui regarde les choses, comme si le faux n'existent pas même dans la pensée, comme si le vrai et les essences, le bien était le vrai »

1. Écrire les fonctions utilitaires suivantes
   - `simplifier : 'a list -> 'a list` qui à une liste d'éléments quelconque retourne une liste en ne conservant qu'une seule occurence de chaque élément ;
   - `successeurs : 'a -> 'a list -> 'a list` telle que `successeurs x l` renvoie la liste des éléments situés après chacune des occurences de `x` dans `l` ;
   - `decouper : string -> string list` qui renvoie la liste formée de la suite des mots du texte dans l'ordre où ils apparaîssent. Vous pouvez vous aider des fonctions prédéfinies `String.contains`, `String.index` et `String.sub`.
2. En vous servant des fonctions précédentes et de `List.map`, écrire la fonction

   ```
   table : string list -> (string * (string list)) list
   ```

   telle que si `lm` est un texte découpé en liste de mots, `table lm` renvoie la table de succession de ce texte
3. Écrire la fonction `piocher` qui renvoie un élément choisi aléatoirement dans une liste. Vous pouvez vous aider de `Random.int`, `List.length` et `List.nth`.
4. Écrire la fonction

   ```
   generer : int -> (string * (string list)) list -> string
   ```

   telle que si `n` est un entier et `t` est une table de succession, `generer n t` renvoie à partir de cette table et suivant la méthode décrite au dessus, un texte aléatoire à `n` mots, séparés par des espaces. Vous pouvez écrire des fonctions auxiliaires.
5. Déduire une fonction

   ```
   spam : int -> string -> string
   ```

   Vous pouvez utiliser la fonction `String.lowercase` afin de convertir un texte en minuscules.

2. *In* Métaph., VI, 4, 1027 b, 25-33, trad., T. I, pp. 344-345