TD machine n°3

Exercice 1. Écrire et typer une fonction `renverser` qui retourne une liste contenant les mêmes éléments qu'une liste `l` donnée en argument mais en ordre inverse.

Écrire deux versions :

— une fonction récursive directe en utilisant l'opérateur `@`,
— une fonction utilisant la fonction `reduction`

Exercice 2. Le codage R.L.E. (Run-Length Encoding)
Le codage R.L.E. est une méthode de compression de listes très simple. Son principe est de remplacer dans une liste une suite de caractères identiques par le couple constitué du nombre de caractères identiques et du caractère. Ainsi, la liste `['a' ; 'a' ; 'a' ; 'b' ; 'a' ; 'b' ; 'b']` est compressée en `[(3,'a') ; (1,'b') ; (1,'a') ; (2,'b')]`.

1. Écrire une fonction de décompression qui prend une liste de données compressées et retourne la liste initiale en utilisant les fonctionnelles `List.map` et `List.flatten`.
2. Écrire une fonction de compression qui prend une liste de données `l` et retourne la liste de données compressées par le codage R.L.E. associée à `l`.

Rappel :

- `List.flatten : 'a list list -> 'a list` : concatenate a list of lists. The elements of the argument are all concatenated together (in the same order) to give the result.
- `map : ('a -> 'b) -> 'a list -> 'b list`
  `List.map f [a1; ...; an] applies function f to a1, ..., an, and builds the list [f a1; ...; f an] with the results returned by f.`

Exercice 3.

1. Écrire une fonction `enum` telle que `enum n` renvoie la liste `[0;1;...;n]`
2. À partir de cette fonction et en utilisant `List.map`, écrire une fonction `enum_droite` telle que `enum_droite i n` renvoie la liste `[(i,0);(i,1);...;(i,n)]`
3. En déduire la fonction `enum_paires : int -> int -> (int * int) list` telle que `enum_paires n p` renvoie la liste des éléments de $\{0, \ldots, n\} \times \{0, \ldots, p\}$ dans l'ordre lexicographique. Exemple : `enum_paires 1 2 = [(0,0);(0,1);(0,2);(1,0);(1,1);(1,2)]`
4. En utilisant la fonction `enum_paires`, écrire une fonction `table_mult n` qui calcule la table de multiplication de 0 à `n`, sous la forme d'une liste de triplets $(i, j, i \times j)$. Exemple :

```
1 table_mult 2 = [(0,0,0);(0,1,0);(0,2,0);
2                 (1,0,0);(1,1,1);(1,2,2);
3                 (2,0,0);(2,1,2);(2,2,4)]
```

Exercice 4. En utilisant uniquement les fonctions de la librairie `List`, écrire une fonction `emballer : 'a list -> 'a list list` qui transforme une liste en une liste de listes à un seul élément. Par exemple, `emballer [1; 2; 3]` renvoie `[ [1]; [2]; [3]]`

Exercice 5. Matrices
On veut manipuler des matrices carrées que l'on va représenter par des listes de listes. la matrice

12	2
27	13

sera représentée par la liste [[12 ;2] ;[27 ;13]]. Sa diagonale est [12 ;13].

1. Écrire une fonction `nieme : 'a list -> int -> 'a` telle que `(nieme l n)` renvoie le n-ième élément de la liste `l` s'il existe (l'élément de tête est 0ième).

2. `diagonale : 'a list list -> 'a list` renvoie la diagonale d'une matrice carrée non vide (on ne se préoccupe pas de ce qui se passe si on l'applique à une matrice vide ou non carrée). Écrire `diagonale`
   — en écrivant puis en utilisant une fonction auxiliaire `diagonale_aux` telle que `diagonale_aux n [l1;l2;...lm])` renvoie le n-ième élément de `l1`, le n+1-ième élément de `l2`, *etc*...
   — directement mais en utilisant uniquement les fonctions suivantes : `map, combine, length, decurryfier` et `from_to`.
3. Écrire une fonction `forall : ('a -> bool) -> 'a list -> bool` telle que `(forall p l)` renvoie `true` si tous les éléments de `l` vérifient le prédicat `p` et renvoie `false` sinon.
4. En utilisant la fonction précédente, écrire une fonction `verifier : 'a list list -> bool` qui vérifie que la matrice en argument est non vide et carrée.