TD 4 : Listes

NB : quelques exercices de ce TD proviennent du site `https://wiki.haskell.org/H-99:_Ninety-Nine_Haskell_Problems` (qui est fortement recommandé)

1 Combinations

Donner une fonction `combinations` qui étant donné un entier `k` et une liste `xs` retourne toutes les sous–listes de `xs` de taille `k`.

Exemple :

```
*Main> combinations 3 "abcde"
["abc","abd","abe","acd","ace","ade","bcd","bce","bde","cde"]
```

Hint : vous pouvez utiliser la fonction `map` ou la compréhension en listes, ou `filter`, ou .... En plus, vous pouvez utiliser la fonction du TD1 qui produit tous les sous-ensembles d'une liste.

2 DropWhile

La fonction `dropWhile` a le type `dropWhile :: (a -> Bool) -> [a] -> [a]`. Cette fonction prend comme argument une condition booléenne et une liste et supprime le préfixe maximum de la liste qui ne contient que des éléments qui satisfont la condition.

Exemple :

```
*Main> dropWhile even [2,4,6,1,3,5,2,4]
[1,3,5,2,4]
```

Donner une implémentation récursive de cette fonction.

3 Compression de listes

Donner une fonction `compress` qui étant donné une liste retourne la même liste avec tous les doublons consécutifs remplacés par une seule copie.

Exemple :

```
*Main> compress [1,2,3,3,2,2,2,5,3,4]
[1,2,3,2,5,3,4]
*Main> compress "aabaabaabbba"
"abababa"
```

1. Donner une implémentation récursive directe.
2. Donner une implémentation qui utilise `dropWhile`.
3. Donner une implémentation qui utilise `filter` et supprime tous les doublons (même non–consécutifs)
   Exemple :

```
*Main> compress' [1,2,3,1,2,3,1,1,2,2,2,3,3,3,1,2]
[1,2,3]
```

4. Donner une implémentation qui utilise `foldr`.
5. Donner une implémentation qui utilise `foldl`.

4 concatMap

La fonction `concatMap` a comme type `concatMap :: (a -> [b]) -> [a] -> [b]`. Étant donné une liste d'éléments de type `a`, et une fonction qui s'applique sur de tels éléments et retourne une liste de `b`, applique cette fonction sur chaque élément et retourne la concaténation de tous les résultats.

Exemple :

```
*Main> concatMap show [12,34,56]
"123456"
```

Donner une implémentation récursive de cette fonction

5 Réplication

Écrire une fonction `replic` qui étant donné une liste `xs` et un entier `n` retourne la même liste où chaque élément est répété `n` fois.

Exemple :

```
*Main> replic  [1..5] 5
[1,1,1,1,1,2,2,2,2,2,3,3,3,3,3,4,4,4,4,4,5,5,5,5,5]
```

1. Donner une implémentation récursive directe. Vous pouvez utiliser les fonction `take` et `repeat`.
2. Donner une implémentation qui utilise `concatMap`.
3. Donner une implémentation qui utilise `foldr`.