Exercices INF8541: série #15

1. Tri rapide `quicksort`

Définissez une fonction de tri `quicksort`. Cette fonction doit recevoir en argument une fonction `comp`, laquelle détermine si deux valeurs sont ordonnées ou non

```
triCroissant   = quicksort (<=)
triDecroissant = quicksort (>=)

triCroissant   [10, 2, 15, 8] == [2, 8, 10, 15]
triDecroissant [10, 2, 15, 8] == [15, 10, 8, 2]
```

2. Quantificateurs

Définissez les quantificateurs logiques suivants:

- `some p l`: Détermine si il existe un élément de `l` qui possède la propriété `p`.
- `all p l`: Détermine si tous les un éléments de `l` possèdent la propriété `p`.
- `unique p l`: Détermine si il existe un et un seul élément de `l` qui possède la propriété `p`.

3. Sous-séquences et entrelacements

1. Définissez une fonction `sousSequences` qui retourne toutes les sous-séquences d'une séquence donnée. Par exemple:

   ```
   sousSequences "era"
   == ["", "a", "r", "ra", "e", "ea", "er", "era"]
   ```

2. Définissez une fonction `entrelacements` qui retourne toutes les façons possibles d'insérer un élément dans une liste. Par exemple:

   ```
   entrelacements 'x' "era"
   == ["xera", "exra", "erxa", "erax"]
   ```

4. Fonction `iterate`

Soit la fonction `iterate` définie informellement comme suit:

```
iterate f x = [x, f x, f (f x), f (f (f x)), ...]
```

1. Donnez une définition de `iterate`.
2. Étant donné un nombre `n`, que calcule l'expression suivante appliquée à ce nombre?

   ```
   reverse . map (`mod` 10) . takewhile (> 0) . iterate (`div` 10)
   ```

   Note: `reverse [x1, x2, ..., xn] == [xn, ..., x2, x1]`

5. Problème de Jackson

Le problème de Jackson, que nous avons examiné précédemment, consiste à transformer des lignes de `N1` caractères en lignes de `N2` caractères tout en changeant les suites de deux caractères "`**`" par un unique "`^`". Développez une solution fonctionnelle à ce problème. Vous pouvez utiliser les types suivants et en-têtes suivants:

```
type LigneN1 = [Char]
type LigneN2 = [Char]

transformerN1N2 :: [LigneN1] -> [LigneN2]
```