TD 2 : Algorithmes de tri

1 Quicksort – Tri rapide

Pendant le cours on a vu l'implémentation de l'algorithme de tri rapide en Haskell :

```
qsort [] = []
qsort (x:xs) = (qsort left) ++ [x] ++ (qsort right)
  where
    left = onlySmaller x xs
    right = onlyLarger x xs

onlySmaller i [] = []
onlySmaller i (x:xs)
  | x < i = x: (onlySmaller i xs)
  | otherwise = onlySmaller i xs

onlyLarger i [] = []
onlyLarger i (x:xs)
  | x > i = x: (onlyLarger i xs)
  | otherwise = onlyLarger i xs
```

Cette implémentation est correcte seulement si la liste donnée ne contient pas de doublons (pourquoi ?). Donner une implémentation qui fonctionne correctement dans tous les cas.

NB : n'oubliez pas de donner les types de toutes vos fonctions.

2 Mergesort – Tri fusion

Rappel : L'algorithme de tri fusion, étant donné un tableau $A[1 \dots n]$ fait les étapes suivantes :

1. Trie la première moitié de A (appel récursif)
2. Trie la deuxième moitié de A (appel récursif)
3. Fusionne les deux tableaux triés

Donner une implémentation de cet algorithme en Haskell. Vous pouvez utiliser les fonctions `take, drop`.

3 InsertionSort – Tri par insertion

Rappel : l'algorithme de tri par insertion trie un tableau $A[1 \dots n]$ comme suit :

1. On trie le tableau $A[2 \dots n-1]$.
2. On insère l'élément $A[1]$ dans la bonne position dans le tableau trié.

Donner une implémentation de cet algorithme en Haskell.

4 SelectionSort – Tri par sélection

Rappel : l'algorithme de tri par sélection trie un tableau $A[1 \ldots n]$ comme suit :

1. Répète n fois
   — On trouve l'élément minimum parmi les éléments qui ne sont pas encore triés et le retire du tableau.

Donner une implémentation de cet algorithme en Haskell. Hint : Tout d'abord, il faut écrire une fonction `findmin` qui, étant donné une liste, retourne la position de l'élément minimum de la liste. Exemple : `findmin [17,15,11,23] --> 2`.