Segmentation d'une liste

Soit $L = [a_0; a_1; \ldots; a_{n-1}]$ une liste d'entiers et p un nombre entier appelé pivot. « Découper » L selon p consiste à calculer les deux listes, éventuellement vides, L_1 et L_2 telles que :

- L_1 contient tous les éléments a_i de L tels que $a_i < p$;
- L_2 contient tous les éléments a_i de L tels que $a_i \geqslant p$;
- l'ordre relatif des éléments de L est conservé dans L_1 et L_2.

« Découper strictement » L selon p consiste à calculer les listes L_1 définie ci-dessus, L'_2 constituée des éléments de L strictement supérieurs à p avec le même ordre que dans L, et le nombre $N(p)$ d'éléments de L égaux à p.

1) Écrire deux fonctions récursives :

```
découpe        : int list -> int -> (int list) * (int list)
découpe_strict : int list -> int -> (int list) * (int list) * int
```

telles que `découpe` L p retourne le couple (L_1, L_2) et `découpe_strict` L p retourne le triplet $(L_1, L'_2, N(p))$. Tester ces fonctions sur des listes aléatoires. On crée une liste aléatoire de n entiers compris entre a et b par le code suivant :

```
let rec random_list a b n =
  if n = 0 then []
  else (a + random__int(b-a)) :: (random_list a b (n-1))
;;
```

2) Applications. En utilisant les fonctions `découpe` et `découpe_strict` programmer les fonctions suivantes :

a) `est_sr : int list -> bool` qui « dit » si la liste L passée en argument est sans répétitions.

b) `majoritaire : int list -> int * int` qui cherche dans la liste L passée en argument *un* élément x dont le nombre d'occurences, $N(x)$, est maximal et retourne le couple $(x, N(x))$.

c) `tri : int list -> int list` qui retourne la liste triée par ordre croissant contenant les mêmes éléments que son argument L. Si un nombre figure plusieurs fois dans L il devra figurer autant de fois dans la liste résultat.

d) `tri_sr : int list -> int list` qui retourne la liste triée par ordre croissant contenant les mêmes éléments que son argument L. Si un nombre figure plusieurs fois dans L il ne devra figurer qu'une fois dans la liste résultat.

e) `nème : int list -> int -> int` qui prend en argument une liste L et un entier k et retourne le k-ème élément par ordre croissant de L. Cette fonction doit être implémentée directement sans faire appel à `tri`.

f) `disjoint : int list -> int list -> bool` qui « dit » si les deux listes L_1 et L_2 passées en argument sont disjointes.