Colle 5 : Recherche du second plus grand élément dans une liste.

On se propose d'écrire une fonction qui retourne le second plus grand élément d'une liste de nombres distincts non triée. La longueur de la liste est supérieure ou égale à 2.

1 UNE PREMIÈRE MÉTHODE SIMPLE

▷ **Question 1.** Ecrire, en utilisant une fonction auxiliaire récursive, une fonction `second_pge1 : 'a list -> 'a` déterminant le second plus grand élément d'une liste. Cette fonction effectuera, dans le pire des cas, $2n-3$ comparaisons (où n est la longueur de la liste donnée). ◁

2 MÉTHODE "DIVISER POUR RÉGNER"

▷ **Question 2.** Ecrire une fonction récursive `partage : 'a list -> 'a list * 'a list` qui prend une liste `l = [e1; e2; . . . ; en]` comme argument et qui renvoie le couple de listes `(l1,l2)` où `l1` est la liste des éléments d'indices impairs de `l` et `l2` la liste des éléments d'indices pairs. Autrement dit `l1 = [e1; e3; . . .]` et `l2 = [e2; e4; . . . ]`. ◁

▷ **Question 3.** Ecrire une fonction `second_pge2 : 'a list -> 'a`, déterminant le second plus grand élément d'une liste en utilisant la méthode "diviser pour régner". Combien de comparaisons cette fonction effectue-t-elle lorsque $n = 2^p$? ◁

3 MÉTHODE PAR TOURNOIS

On réalise une suite de tournois de la manière suivante :

- La liste `l = [e1; . . . ; en]` est transformée en une liste `[(e1, [ ]);(e2, [ ]); . . . ; (en, [ ])]`
- On partage la liste `l = [(e1, [ ]);(e2, [ ]); . . . ; (en, [ ])]` en deux listes

  `l1 = [(e1, [ ]);(e3, [ ]); . . . ; (e2k+1, [ ])]`

  et `l2 = [(e2, [ ]);(e4, [ ]); . . . ; (e2p, [ ])]` avec $p = k$ ou $p = k+1$.
- Le premier tournoi consiste à comparer le premier élément de `l1` au premier de `l2`, puis le second de `l1` au second de `l2` et ainsi de suite et de construire la liste des résultats `(e01, [e02])` où `e01` est le vainqueur (plus grand) et `e02` le vaincu.
- La liste des résultats est alors scindée en deux listes pour l'organisation du second tournoi et ainsi de suite jusqu'à obtenir une liste de résultats comprenant un seul couple `(e, [f1, . . . , fm])` : en toute généralité le résultat du match de `(e, [f1, . . . , fm])` avec `(e0, [f01, . . . , f0m])` est `(e, [e0, f1, . . . , fm])` si `e > e0` ; `(e0, [e, f01, . . . , f0m])` sinon.

Le second plus grand élément est le plus grand élément de la liste finale des vaincus.
Par exemple considérons `l = [2; 3; 1; 0; 4; 2; 7; 9; 8] ;` alors
`l1 = [(2, [ ]);(1, [ ]);(4, [ ]);(7, [ ]);(8, [ ])]` et `l2 = [(3, [ ]);(0, [ ]);(2, [ ]);(9, [ ])]`.
Le résultat du premier tournoi donne `l0 = [(3, [2]); (1, [0]); (4, [2]); (9, [7]); (8, [ ])]`.
Le partage de `l0` donne `l01= [(3, [2]); (4, [2]); (8, [ ])]` et `l02= [(1, [0]); (9, [7])]`.

Le résultat du second tournoi donne `l00 = [(3, [1; 2]); (9, [4; 7]); (8, [ ])]`
Le partage de `l00` donne `l001 = [(3, [1; 2]); (8, [ ])]` et `l002 = [(9, [4; 7])`
Le résultat du troisième tournoi donne `l000 = [(9, [3; 4; 7]); (8, [ ]]`, et enfin un dernier tournoi donne `[(9, [8; 3; 4; 7])]`
Pour obtenir le second plus grand élément on extrait alors le plus grand élément des vaincus. Dans l'exemple ci-dessus, le second plus grand élément est 8 (le plus grand élément de la liste `[8; 3; 4; 7]`).

▷ **Question 4.** Justifiez cet algorithme. ◁

▷ **Question 5.** Si l'on suppose que `l` contient $n = 2^k$ éléments, déterminer le nombre de tours effectués, puis le nombre total de comparaisons entre les éléments de la liste `l`. ◁

▷ **Question 6.** Ecrire une fonction récursive `un_tour : ('a * 'a list) list -> ('a * 'a list) list -> ('a` qui prend deux listes `l1` et `l2` d'éléments de la forme `(i, advi)`, où `advi` est la liste des adversaires de i jusqu'alors et qui renvoie la liste des gagnants avec leurs adversaires rencontrés jusqu'à présent. ◁

▷ **Question 7.** Ecrire une fonction récursive `tournoi : ('a * 'a list) list -> 'a * 'a list` qui prend une liste de couples éléments et liste d'adversaires, et qui renvoie le couple formé du gagnant et des adversaires qu'il a rencontrés. ◁

▷ **Question 8.** Avant le premier tournoi, la liste des adversaires de chaque élément est vide ; écrire une fonction récursive `prepare_tournoi : 'a list -> ('a * 'b list) list` qui prend la liste des éléments à comparer et qui renvoie la liste `[(e1, []), . . . , (en, [])]`. ◁

▷ **Question 9.** Ecrire une fonction récursive `pge` qui renvoie le plus grand élément d'une liste, puis la fonction finale `second_pge : 'a list -> 'a` qui renvoie le second plus grand élément d'une liste. ◁

4 Bonus

▷ **Question 10.** ◁

1.2 Recherche de l'élément de rang k d'une suite de valeurs dans deux cas particuliers

Nous considérons une suite de n valeurs parmi lesquelles nous cherchons l'élément de rang k ; c'est-à-dire l'élément d'indice k une fois ces éléments classés par ordre croissant (le premier élément par ordre croissant ayant pour indice 1).

Par exemple si on considère la suite de valeurs $(3, 7, 2, 4, 3, 5, 1, 1, 9)$. Alors classée par ordre croissant cette suite devient $(1, 1, 2, 3, 3, 4, 5, 7, 9)$; l'élément de rang 8 est 7. Il n'y a pas d'élément de rang 10.

Nous écrirons la fonction rang dans deux cas simples : la suite des valeurs est donnée par une liste classée en ordre croissant, la suite des valeurs est la réunion de deux listes, chacune étant classée en ordre croissant. Autrement dit :

Question 7. Ecrire une fonction récursive `rg : int -> 'a list -> int` telle que, si `l` est une liste classée en ordre croissant `rg k l` renvoie l'élément de rang `k` de la suite de valeurs de cette liste.

Par exemple `rg 3 [1;1;4;5;7] -> 4`

`rg 6 [1;1;4;5;7] -> Uncaught exception: Failure "inexistant"`

Question 8. Ecrire une fonction récursive `rang_l : int-> 'a list * 'a list -> 'a` telle que, si `l1` et `l2` sont deux listes triées par ordre croissant et `k` est un entier tel que $1 \leqslant$ `k` $\leqslant$`|l1|+|l2|`, alors `rang_l k (l1,l2)` renvoie l'élément de rang k de la suite de valeurs correspondant à l'union des deux listes.

Par exemple `rang_l 5 ([2;2;7;8;15;20],[2;5;12]) -> 7`

La fonction devra effectuer au plus $(n-1)$ comparaisons, où n=`|l1|+|l2|`. S'il n'existe pas de valeur de rang `k`, une exception affichant `inexistant` est déclenchée.

Pour « tester » votre fonction vous pouvez utiliser l'exemple précédent et l'exemple, `rang_l 5 ([2;4],[3;4;5;7;9])`