Exercice 1.
Ecrire une fonction `listeN(n)` qui reçoit en argument un entier positif ou nul `n` et renvoie la liste contenant les entiers `1`, `2`, ..., `n` dans cet ordre. Si `n` = 0, la liste renvoyé en vide.

Exercice 2.
Ecrire une fonction `affiche_liste(lst)` qui affiche, en utilisant la fonction `print`, tous les élément de la liste `lst`, séparés par des espaces. L'écrire comme une fonction récursive, puis avec une boucle `while`.

Exercice 3.
Réécrire la fonction `nieme_element` avec une boucle `while`.

Exercice 4.
Ecrire une fonction `occurences(x, lst)` qui renvoie le nombre d'occurences de `x` dans `lst`. L'écrire comme une fonction récursive, puis avec une boucle `while`.

Exercice 5.
Ecrire une fonction `trouve(x, lst)` qui renvoie le rang de la première occurence de `x` dans `lst`, le cas échéant, et `None` sinon. L'écrire comme une fonction récursive, puis avec une boucle `while`.

Exercice 6.
Ecrire une fonction `identique(l1, l2)` qui renvoie un booléen indiquant si les listes `l1` et `l2` sont identiques, c'est-à-dire contiennent exactement les mêmes éléments, dans le même ordre. On suppose que l'on peut comparer les éléments de `l1` et `l2` avec l'égalité `==` de Python.

Exercice 7.
Ecrire une fonction `inserer(x, lst)` qui prend en arguments un entier `x` et une liste d'entiers `lst`, supposée triée par ordre croissant, et qui renvoie une nouvelle liste dans laquelle `x` a été inséré à sa place. Ainsi, insérer la valeur `3` dans la liste 1, 2, 5, 8 renvoie la liste 1, 2, 3, 5, 8.

Exercice 8.
En se servant de l'exercice précédent, écrire une fonction `tri_par_insertion(lst)` qui prend en argument une liste d'entiers `lst` et renvoie une nouvelle liste, contenant les mêmes éléments et triée par ordre croissant. On suggère de l'écrire comme une fonction récursive.

Exercice 9.
Ecrire une fonction `liste_de_tableau(t)` qui renvoie une liste qui contient les éléments du tableau `t`, dans le même ordre. On suggère de l'écrire avec une boucle `for`.

Exercice 10.
Ecrire une fonction `derniere_cellule(lst)` qui renvoie la dernière cellule de la liste `lst` et `None` si `lst` est vide.

Exercice 11.
En utilisant la fonction de l'exercice précédent, écrire une fonction `concatener_en_place(l1, l2)` qui réalise la concaténation en place des listes `l1` et `l2`, c'est-à-dire qui relie la dernière cellule de `l1` à la première de `l2`. Cette fonction doit renvoyer la toute première cellelue de la concaténation.