Exercice 1

Écrire une fonction `separe` qui prend en paramètres, une liste `T` de nombres entiers relatifs non triés, un entier relatif `borneInf` et entier relatif `bornesup` et qui renvoie trois tuples, le premier `t1` contenant tous les éléments de `T` strictement inférieurs à `borneInf` , le deuxième `t2` contenant tous les éléments de `T` strictement supérieurs à `borneSup`, et le troisième `t3` les éléments restant.

Par exemple :

```
separe([5, 12, -9, 0, 3, -2, 7, -1], -4, 4)
```

renvoie

```
((-9,), (5, 12, 7), (0, 3, -2, -1))
```

Écrire une documentation (docstring) et un jeu de tests pour cette fonction.

Exercice 2

Soit `t` un tuple de nombres entiers rangés par ordre croissant. On veut insérer un entier `n` dans ce tuple. Si `n` existe déjà dans `t`, il est ignoré. Compléter le code de la fonction récursive `insere`.

On rappelle que pour construire un tuple vide, il est possible d'utiliser la fonction native : `x = tuple()` ou plus simplement d'écrire : `x = ()`, et que pour construire un tuple ne comprenant qu'une seule valeur il suffit d'écrire : `x = (5,)` ou plus simplement `x = 5,`

```
In [1]: def insere(t, n, i=0):
    ...:
    ...:     if n < t[0]:
    ...:         return ??? + t
    ...:
    ...:     if n > t[???]:
    ...:         return ???
    ...:
    ...:     if i == len(t):
    ...:         return ???
    ...:
    ...:     if t[i] < n and n < t[i + ???]:
    ...:         return (???, n) + insere(t, n, i+1)
    ...:
    ...:     return ??? + insere(t, n, i+1)

In [2]: insere((2,3,6,8,9), 4)
Out[2]: (2, 3, 4, 6, 8, 9)

In [3]: insere((2,3,6,8,9), 1)
Out[3]: (1, 2, 3, 6, 8, 9)

In [4]: insere((2,3,6,8,9), 12)
Out[4]: (2, 3, 6, 8, 9, 12)

In [5]: insere((2,3,6,8,9), 6)
Out[5]: (2, 3, 6, 8, 9)

In [6]: insere((2,3,6,8,9), 9)
Out[6]: (2, 3, 6, 8, 9)

In [7]: insere((2,3,6,8,9), 2)
Out[7]: (2, 3, 6, 8, 9)
```