Complément au corrigé du DM d'ITC n°1

Q24 – On donne ici deux versions de la fonction `fusion`, sans utiliser de méthodes exotiques comme `pop(0)` (qui est d'ailleurs très peu efficace en terme de complexité temporelle) ou encore `extend` (pas au programme).

Version avec butée :

```
1  def fusion(L1,L2) :
2    n1, n2 = len(L1), len(L2)
3    L = (n1+n2)*[0]   # initialisation de L
4    butee = -1   # valeur plus petite que toutes celles de L1 et L2
5    L1.append(butee)
6    L2.append(butee)
7    i, j = 0, 0
8    for k in range(n1+n2) :
9      if L1[i] >= L2[j] :
10       L[k] = L1[i]    # on met le plus grand d'abord dans L
11       i += 1
12     else :
13       L[k] = L2[j]
14   return L
```

Remarque :

On pourrait aussi remplacer la ligne 3 par `L = []` et donc la ligne 10 par `L.append(L1[i])` et la ligne 13 par `L.append(L2[i])`

Voici maintenant une version sans butée :

```
1  def fusion(L1,L2) :
2    n1, n2 = len(L1), len(L2)
3    L = (n1+n2)*[0]
4    k = 0
5    while (k < n1) and (k < n2) :
6      if L1[k] >= L2[k] :
7        L[k] = L1[k]
8      else :
9        L[k] = L2[k]
10     k += 1
11   if k < n1 :
12     L = L + L1[k : ]
13   elif k < n2 :
14     L = L + L2[k : ]
15   return L
```

Remarque :

À la ligne 13 on peut remplacer `elif k < n2` par un simple `else` puisque l'une des deux listes `L1` ou `L2` a déjà été parcourue jusqu'au bout dans la boucle `while` : il suffit ensuite de "vider" ce qui reste de l'autre liste dans `L`.

Q25 – On donne un version qui ne fait appelle qu'une fois à `len(L)` (ce qui réduit le temps d'exécution de la fonction).

```
1  def trier_facettes (L):
2    n = len(L)
3    if n <= 1:
4      return L
5    else :
6      m = n//2
7      L1 = trier_facettes (L[ : m])
8      L2 = trier_facettes (L[m : ])
9      return fusion (L1,L2)
```