Correction du DS 0

Informatique pour tous, deuxième année

Julien Reichert

Exercice 1

```
def den_grimme_aelling(liste):
    ll = []
    for element in liste:
        if element > 0:
            ll.append(1)
        elif element < 0:
            ll.append(-1)
        else:
            ll.append(0)
    return ll
```

Exercice 2

```
def un_commun(l, ll):
    n, nn = len(l), len(ll)
    i, ii = 0, 0
    while i < n and ii < nn:
        if l[i] == ll[ii]:
            return True
        elif l[i] < ll[ii]:
            i += 1
        else:
            ii += 1
    return False
```

Exercice 3

On commence par prouver la terminaison : `nn + n - (ii + i)` est un variant qui diminue d'un à chaque tour de boucle. Ceci permet de déduire que le nombre de tours de boucle est majoré par la somme des tailles des listes, et que la complexité est linéaire en cette somme puisque le corps de boucle est en $\mathcal{O}(1)$.

Exercice 4

Le plus grand produit est le maximum entre le produit des deux plus petits nombres et celui des deux plus grands nombres. Il ne peut être strictement négatif que s'il n'y a que deux éléments dans la liste, tous non nuls et de signe opposé. On cherche donc ces quatre (ou moins si la liste est trop courte) valeurs, le tout en un seul passage.

Pour information, il existe une optimisation (qui reste en temps linéaire mais avec un meilleur facteur constant) de la recherche des deux plus grands éléments, cette optimisation n'est pas traitée ici.

```
def plus_grand_produit(l):
    assert len(l) >= 2, "Liste trop courte"
    max1 = l[l[0] < l[1]] # Oh la belle astuce pour éviter le copier-coller !
    max2 = l[l[0] > l[1]]
    min1 = l[l[0] > l[1]]
    min2 = l[l[0] < l[1]]
    for i in range(2, len(l)):
        if l[i] > max1:
            max2 = max1
            max1 = l[i]
        elif l[i] > max2:
            max2 = l[i]
        if l[i] < min1: # pas elif
            min2 = min1
            min1 = l[i]
        elif l[1] < min2:
            min2 = l[i]
    return max(max1 * max2, min1 * min2)
```

Exercice 5

```
def crible(n):
    reponse = [True for i in range(n+1)]
    premiers = []
    for i in range(2, n+1):
        if reponse[i]:
            premiers.append(i)
            for j in range(2*i, n+1, i):
                reponse[j] = False
    return premiers
```

Exercice 6

```
def quart_de_tour(l):
    return [[l[-1-i][j] for i in range(len(l))] for j in range(len(l[0]))]
```

Exercice 7

```
def grenouille(l):
    esnoper = ""
    l_ensemble = set(l)
    ecidni = l[-1]
    while ecidni > 0:
        if ecidni - 2 in l_ensemble:
            esnoper += "2"
            ecidni -= 2
        else:
            esnoper += "1"
            ecidni -= 1
    return esnoper[::-1]
```