

# Algorithmique

## Correction Partiel n° 1 (P1)

INFO-SUP S1 – EPITA

7 janvier 2020 - 13h-15h

**Solution 1 (Types Abstraits : liste itérative (supprimer) – 3 points)**

### OPÉRATIONS

*supprimer* : liste  $\times$  entier  $\rightarrow$  liste

où *supprimer*( $\lambda$ ,  $k$ ) supprime le  $k^{\text{ème}}$  élément de la liste  $\lambda$  ( $k$  commence à 1).

1. On peut mettre la précondition suivante, ce qui "allégera" les axiomes. *Ne pas la préciser ici n'est pas grave si le postulat existe au niveau des axiomes.*

### PRÉCONDITIONS

*supprimer*( $\lambda$ ,  $k$ ) **est-défini-ssi**  $\lambda \neq \text{liste-vide}$  &  $1 \leq k \leq \text{longueur}(\lambda)$

La première condition  $\lambda \neq \text{liste-vide}$  peut être omise puisqu'elle est "incluse" dans la deuxième !

2. Les axiomes sont les suivants :

### AXIOMES

$\text{longueur}(\text{supprimer}(\lambda, k)) = \text{longueur}(\lambda) - 1$

$1 \leq i < k \Rightarrow \text{ième}(\text{supprimer}(\lambda, k), i) = \text{ième}(\lambda, i)$

$k \leq i \leq \text{longueur}(\lambda) - 1 \Rightarrow \text{ième}(\text{supprimer}(\lambda, k), i) = \text{ième}(\lambda, i + 1)$

### AVEC

$\lambda$  : Liste

$k, i$  : Entier

**Solution 2 (Dichotomie : "chemin" de recherche – 2 points)**

	NON	OUI
46 - 65 - 81 - 73 - 70 - 66		✓
31 - 62 - 90 - 72 - 61 - 66	✓	
36 - 70 - 53 - 40 - 42 - 66	✓	
35 - 51 - 55 - 58 - 61 - 66		✓

**Solution 3 (Test - )**

`test(x, L)` vérifie si  $x$  est présent dans la liste  $L$ .

---

**Solution 4 (Entiers  $\leftrightarrow$  liste – 6 points)**

1. La fonction `int_to_list(n, p)` retourne la liste des  $p$  chiffres de  $n$  :

```
1      def int_to_list(n, p):
2          L = []
3          while n != 0:
4              L.append(n % 10)
5              n = n // 10
6          for i in range(p-len(L)):
7              L.append(0)
8          return L
9
10     #
11
12     def int_to_list2(n, p):
13         L = []
14         while p > 0:
15             L.append(n % 10)
16             n = n // 10
17             p -= 1
18         return L
```

2. La fonction `list_to_ints([d1, d2, ..., dp])` retourne le couple d'entiers  $(d_1d_2 \dots d_p, d_p \dots d_2d_1)$  :

```
1      def list_to_ints(L):
2          left = 0
3          right = 0
4          n = len(L)
5          for i in range(n):
6              left = left * 10 + L[i]
7              right = right * 10 + L[n-i-1]
8          return (left, right)
9
10     #
11
12     def list_to_ints2(L):
13         left = 0
14         right = 0
15         p = 1
16         for i in range(len(L)):
17             left = left * 10 + L[i]
18             right = right * 10 + L[i]*p
19             p = p * 10
20         return (left, right)
```

**Solution 5 (Histogramme et tri – 4 points)**

1. La fonction `hist(L)` retourne la liste représentant l'histogramme des valeurs de  $L$  ( $L$  ne contient que des chiffres) :

```
1
2     def hist(L):
3         H = []
4         for i in range(10):
5             H.append(0)
6         # ou H = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
7         for e in L:
8             H[e] += 1
9         return H
```

2. La fonction `sort(L)` retourne la liste  $L$  triée en ordre croissant ( $L$  ne contient que des chiffres) :

```
1
2     def sort(L):
3         H = hist(L)
4         L = []
5         for i in range(10):
6             for nb in range(H[i]):
7                 L.append(i)
8         return L
```

**Solution 6 (Kaprekar – 5 points)**

La fonction `Kaprekar(n, p)` applique le procédé de Kaprekar à  $n$ , entier positif de  $p$  chiffres, jusqu'à ce qu'une valeur soit rencontrée deux fois. Elle affiche les différentes valeurs calculées.

```
1     def Kaprekar(n, p):
2         L = []
3         while not test(n, L):
4             L.append(n)
5             digits = int_to_list(n, p)
6             digits = sort(digits)
7             (low, high) = list_to_ints(digits)
8             n = high - low
9         L.append(n)
10        return L
```