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 × entier → liste

où *supprimer(λ, k)* supprime le $k^{ème}$ élément de la liste λ (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(λ, k) **est-défini-ssi** $\lambda \neq$ *liste-vide* & $1 \leqslant i \leqslant$ *longueur(l)*

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

2. Les axiomes sont les suivants :

   AXIOMES

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

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

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

   AVEC

   λ : 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 :

```
            def int_to_list(n, p):
                L = []
                while n != 0:
                    L.append(n % 10)
                    n = n // 10
                for i in range(p-len(L)):
                    L.append(0)
                return L

#-------------------------------------------------------------------------------

            def int_to_list2(n, p):
                L = []
                while p > 0:
                    L.append(n % 10)
                    n = n // 10
                    p -= 1
                return L
```

2. La fonction `list_to_ints(`$[d_1, d_2, \cdots, d_p]$`)` retourne le couple d'entiers $(d_1 d_2 \cdots d_p,\ d_p \cdots d_2 d_1)$:

```
            def list_to_ints(L):
                left = 0
                right = 0
                n = len(L)
                for i in range(n):
                    left = left * 10 + L[i]
                    right = right * 10 + L[n-i-1]
                return (left, right)

#-------------------------------------------------------------------------------

            def list_to_ints2(L):
                left = 0
                right = 0
                p = 1
                for i in range(len(L)):
                    left = left * 10 + L[i]
                    right = right + L[i]*p
                    p = p * 10
                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
```