

Algorithmique

Correction Partiel n° 1 (P1)

INFO-SUP (S1) – EPITA

3 Jan. 2017 - 10 : 00

Solution 1 (Piles et autres... – 3 points)

1. (a) Est-ce que les séquences suivantes sont valides ?
 - i. Oui!
 - ii. Non : La quatrième sortie intervient sur un garage (pile) vide, en effet v_1, v_2 et v_3 sont déjà sorties.
 - (b) **La règle :**

Une séquence formée de 'E1' de 'E2' et de 'D' est dite *admissible* si elle contient autant de 'D' que de 'E1' et de 'E2' cumulés et si toutes les actions qui lui correspondent peuvent être accomplies dans l'ordre indiqué par la séquence : on ne peut dépiler que s'il reste au moins un élément. A la fin, la pile (garage) doit être vide.
-

Solution 2 (ABR : chemin de recherche – 2 points)

Les séquences ② et ③ sont impossibles :

- ① 46, on part à droite - 65, on part à droite - 81 on part à gauche - 73, on part à gauche - 70, on part à gauche - **66**
- ② 31, on part à droite - **62, on part à droite** - 90, on part à gauche - 72, on part à gauche - **61 ne peut se trouver là, il n'est pas supérieur à 62**
- ③ 36, on part à droite - 70, on part à gauche - **53, on part à droite - 50, ne peut se trouver là, il n'est pas supérieur à 53**
- ④ 35, on part à droite - 51, on part à droite - 55 on part à droite - 58, on part à droite - 61, on part à droite - **66**

Solution 3 (Test - 1 point)

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

Solution 4 (Entiers \leftrightarrow liste – 5 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 + 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
3         L = []
4
5         while not test(n, L):
6             print(n, end=' -> ')
7
8             L.append(n)
9
10            digits = int_to_list(n,p)
11            digits = sort(digits)
12            (low, high) = list_to_ints(digits)
13            n = high - low
14
15            print(n)
```