Algorithmique
Correction Partiel n° 1 (P1)

INFO-SUP S1# – EPITA

20 Juin 2019

Solution 1 **(Pile ou file ? –** ***2 points*****)**

	pile	file	aucune
A B C D E F	√	√	
B D E F A C			√

	pile	file	aucune
D E C B F A	√		
F E D C B A	√		

Solution 2 **(Algorithmes de recherches -** ***3 points*****)**

	Recherche séquentielle			*Recherche dichomique*		
	coût = 1	coût maximum		coût = 1	coût maximum	
	valeur ?	valeur ?	coût ?	valeur ?	valeur ?	coût ?
(a) $n = 20$	u_0	u_{19}	20	u_9 ou u_{10}	u_0	5 ou 9
(b) $n = 100$	u_0	u_{99}	100	u_{49} ou u_{50}	u_0	7 ou 13

Solution 3 **(Croissant – *3 points*)**

Écrire la fonction `is_sorted(L)` qui vérifie si les éléments de la liste L passée en paramètre sont rangés dans l'ordre croissant.

```
1
2  def is_sorted(L) :
3    i = 0
4    n = len(L) - 1
5    while i < n and L[i] <= L[i+1]:
6      i += 1
7    return i >= n
```

Solution 4 **(Tri fusion – *10 points*)**

1. **Spécifications :**
   La fonction `partition` (L) sépare la liste L une liste en deux listes de longueurs quasi identiques (à 1 près) : une moitié dans chaque liste.

```
1          def partition(L):
2
3              n = len(L)
4              L1 = []
5              for i in range(0, n//2):
6                  L1.append(L[i])
7
8              L2 = []
9              for i in range(n//2, n):
10                 L2.append(L[i])
11
12             return (L1, L2)
```

2. **Spécifications :**
   La fonction `merge`$(L1, L2)$ fusionne deux listes $L1$ et $L2$ triées en ordre croissant en une seule liste triée.

```
1          def merge(L1, L2):
2
3              R = []
4              i = j = 0
5              n1 = len(L1)
6              n2 = len(L2)
7
8              while (i < n1) and (j < n2):
9                  if L1[i] <= L2[j]:
10                     R.append(L1[i])
11                     i = i+1
12                 else:
13                     R.append(L2[j])
14                     j = j+1
15
16             for i in range(i, n1):
17                 R.append(L1[i])
18             for j in range(j, n2):
19                 R.append(L2[j])
20
21             return R
```

3. **Spécifications :**
   La fonction `sort`(L) trie la liste L en ordre croissant (pas en place : la fonction construit une nouvelle liste qu'elle retourne).

```python
def mergesort(L):

    if len(L) <= 1:
        return L

    else:
        (L1, L2) = partition(L)

        return merge(sort(L1), sort(L2))
```

Solution 5 **(What is it ? –** ***3 points*****)**

1. Valeur de la liste L après application de `what(`L`,` x`)` :
   (a) L = `[1, 3, 4, 5, 6, 7]`
   (b) L = `[1, 1, 1, 2, 2, 4, 5, 5, 5]`
   (c) L = `[1, 1, 1, 2, 2, 3, 3, 3, 3, 4]`
   (d) L = `[1, 3, 5, 7, 9]`
2. La fonction `what(`L`,` x`)` supprime toutes les valeurs de x dans la liste triée L.