

# Algorithmique

## Partiel n° 1 (P1)

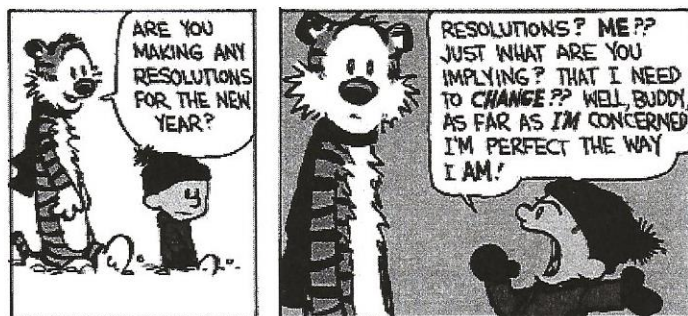
INFO-SUP S1  
EPITA

3 janvier 2023 - 9h30

---

### Consignes (à lire) :

- ☐ Vous devez répondre sur **les feuilles de réponses prévues à cet effet**.
    - Aucune autre feuille ne sera ramassée (gardez vos brouillons pour vous).
    - Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées** : utilisez des brouillons !
    - Ne séparez pas les feuilles à moins de pouvoir les ré-agrafer pour les rendre.
    - Aucune réponse au crayon de papier ne sera corrigée.
  - ☐ La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
  - ☐ **Le code :**
    - Tout code doit être écrit dans le langage Python (pas de C, CAML, ALGO ou autre).
    - **Tout code Python non indenté ne sera pas corrigé.**
    - Tout ce dont vous avez besoin (fonctions, méthodes) est indiqué en **annexe** !
  - ☐ Durée : 2h00
- 



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

On ajoute, dans cet ordre, les valeurs  $A, B, C, D, E$  et  $F$  à une structure linéaire vide. Pour chacun des ordres de sortie donnés, indiquer si la structure en question peut être : une pile, une file (ce peut être les deux), ou aucune des deux (ni une pile, ni une file).

$A B C D E F$   
 $F E D C B A$   
 $C B E F D A$   
 $C E D F A B$

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

Supposons des listes d'entiers triées en ordre croissant. On effectue dans celles-ci une recherche dichotomique de la valeur 42. Parmi les séquences suivantes, lesquelles pourraient correspondre à la suite des valeurs rencontrées lors de la recherche ?

- ① 58 - 33 - 46 - 43 - 39 - 42
- ② 40 - 75 - 57 - 53 - 44 - 42
- ③ 51 - 43 - 34 - 49 - 41 - 42
- ④ 61 - 17 - 38 - 46 - 35 - 42

**Exercice 3 (Lucas – 3 points)**

Écrire une fonction `check_lucas(L)` qui vérifie si chaque élément de la liste d'entiers  $L$  est la somme des deux précédents, les valeurs des deux premiers éléments pouvant être quelconques.

*Exemples d'applications :*

```
1 >>> check_lucas([])
2 True
3 >>> check_lucas([-2, -1, 0, 1, 2])
4 False
5 >>> check_lucas([-2, -1, -3, -4, -7])
6 True
7 >>> check_lucas([0, 1, 1, 2, 3, 5, 8])
8 True
```

#### Exercice 4 (Split – 4 points)

Écrire une fonction `my_split(L, sep)`, avec `L` une liste et `sep` une valeur de "séparation" du type des éléments de `L`. Elle retourne une liste contenant les nombres non nuls de valeurs dans `L` :

- avant la première valeur `sep` (ou jusqu'à la fin de la liste si pas de `sep`),
- entre chaque paire de valeurs `sep`,
- après la dernière valeur `sep`.

Exemples d'applications :

```
1 >>> my_split([], 42)
2 []
3 >>> my_split([1, 2, 3], 42)
4 [3]
5 >>> my_split([-1, 12, 5, -1, 2, 8, -1, -4, 1, 42, -4, -4, -1], -1)
6 [2, 2, 5]
7 >>> my_split(['|', '|', 'a', 'r', '|', '|', '|', 'z', 'e', 'e', '|', '|'], '|')
8 [2, 3]
9 >>> my_split(['u', '%', 'z', 's', '%', 'o', 'k', '.'], '%')
10 [1, 2, 3]
```

#### Exercice 5 (Insertion somme – 5 points)

Écrire la fonction `insert_sum(L, s, x)` qui insère l'élément `x` dans la liste d'entiers `L` à la position suivante : la première position dans `L` telle que la somme des éléments consécutifs de `L` jusqu'à cette position est strictement plus grande que le paramètre `s`.

S'il n'est pas possible d'insérer l'élément `x`, la fonction ne modifie pas la liste. On suppose que la liste contient uniquement des éléments positifs ou nuls et que le paramètre `s` est strictement positif.

Les modifications de la liste `L` devront être effectuées **en place** (voir exemples).

Exemples d'applications :

```
1 >>> L = []
2 >>> insert_sum(L, 1, 42)
3 >>> L
4 []
5 >>> L = [10, 4]
6 >>> insert_sum(L, 5, 42)
7 >>> L
8 [10, 42, 4]
9 >>> insert_sum(L, 100, 42)
10 >>> L
11 [10, 42, 4]
12 >>> insert_sum(L, 53, 42)
13 >>> L
14 [10, 42, 4, 42]
```

Soient les fonctions suivantes :

1. Quel est le résultat de l'application de `mystery([6, 24, 30, 22, 29, 20, 29, 8, 40, 7])` ?
2. Quel est le résultat de l'application de `mystery([4, 14, 6, 18, 4, 7, 5, 19, 14, 11, 11, 3, 11, 13, 4])` ?
3. Que retourne la fonction `mystery(L)` ?
4. Quelles sont les hypothèses que la liste `L` doit respecter pour que `mystery(L)` retourne le résultat décrit à la question précédente ?



## Annexe

### Fonctions et méthodes autorisées

Vous pouvez utiliser les méthodes `append` et `pop` (sans paramètre), la fonction `len` sur les listes ainsi que la fonction `range` :

```
1     >>> L = []
2
3     >>> for i in range(5):
4         L.append(i)
5
6     >>> L
7     [0, 1, 2, 3, 4]
8
9     >>> len(L)
10    5
11
12    >>> for i in range(5, 10):
13        L.append(i)
14    >>> L
15    [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
16
17    >>> L.pop()
18    9
```

Aucun opérateur n'est autorisé sur les listes (+, \*, == ...).

### Vos fonctions

Vous pouvez également écrire vos propres fonctions, dans ce cas vous devez donner leurs spécifications : on doit savoir ce qu'elles font.

Dans tous les cas, la dernière fonction écrite doit être celle qui répond à la question.