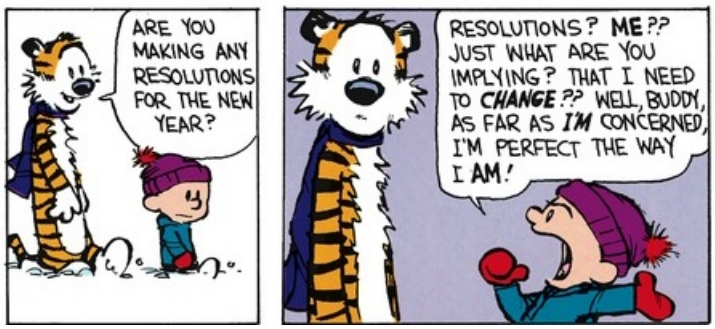
Algorithmique
Partiel n° 1 (P1)

Info-sup S1
Epita

8 Jan. 2019 - 10 : 00

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

Supposons des listes d'entiers triées en ordre croissant. Si on effectuait 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 50 - 15 - 48 - 22 - 46 - 42
② 48 - 15 - 45 - 22 - 47 - 42
③ 15 - 22 - 45 - 43 - 35 - 42
④ 22 - 45 - 43 - 15 - 35 - 42

Exercice 2 (Algorithmes de recherche – *3 points*)

Soit la liste λ suivante : $\lambda = \{1, 5, 7, 16, 21, 30, 33, 34, 42, 49, 72, 81, 99\}$

On cherche la valeur 40 dans cette liste. Pour chaque méthode de recherche, donner le nombre de comparaisons effectuées entre la valeur cherchée et un élément de la liste.

1. Recherche séquentielle sans tenir compte de l'ordre
2. Recherche séquentielle en tenant compte de l'ordre
3. Recherche dichotomique

Exercice 3 (Voir Syracuse – *3 points*)

La suite de Syracuse est définie par ($n \in \mathbb{N}^*$) :

si n est pair, le nombre suivant sera $n/2$
si n est impair, le nombre suivant sera $3n + 1$

Conjecture de Syracuse (ou de Collatz) : $\forall n \in \mathbb{N}^*$ cette suite s'arrête toujours avec la valeur $n = 1$.

Écrire la fonction `Syracuse` qui, à partir de l'entier n, construit la liste contenant les valeurs de la suite de Syracuse jusqu'à obtenir la valeur 1 si n est bien un entier naturel non nul. Dans le cas contraire la fonction retourne une liste vide.

Exemple d'application :

```
>>> Syracuse(14)
[14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1]
```

Exercice 4 (Progression arithmétique – *4 points*)

Écrire une fonction qui vérifie si les valeurs entières d'une liste, contenant au moins deux éléments, suivent une progression arithmétique.

Rappel : Une progression arithmétique est une suite de nombres rangés dans un ordre tel que chacun d'eux s'obtient en ajoutant un nombre constant (la *raison*) à celui qui le précède.

Si la liste a au moins deux éléments et suit bien une progression arithmétique de raison non nulle, la fonction devra retourner la *raison* de la suite. Dans le cas contraire, la fonction devra retourner 0.

Exemples d'applications :

```
>>> arithmetic([1, 2, 3, 4, 5, 6, 7])
1
>>> arithmetic([1, 1, 1, 1, 1])
0
>>> arithmetic([12])
0
>>> arithmetic([5, 8, 11, 14, 17])
3
```

Exercice 5 (Suppression dans liste triée – *5 points*)

Écrire la fonction `delete(`L`, `x`)` qui supprime la valeur x, si elle existe, dans la liste L strictement croissante. La fonction devra retourner un booléen indiquant si la suppression a pu être effectuée.

Exemples d'applications :

```
>>> L = [1, 4, 5, 7, 8, 12, 15]
>>> delete(L, 7)
True
>>> L
[1, 4, 5, 8, 12, 15]
>>> delete(L, 11)
False
>>> L
[1, 4, 5, 8, 12, 15]
```

Exercice 6 (What is it ? – *3 points*)

Soit la fonction `what` ci-dessous :

```
def what(L) :
    me = 0
    for e in L :
        if e > me :
            me = e

    X = []
    for i in range(me + 1) :
        X.append(0)
    for e in L :
        X[e] +=1

    LT = []
    for i in range(me + 1) :
        for j in range(X[i]):
            LT.append(i)
    return LT
```

1. Donner le résultat de l'application suivante de `what` :

```
>>> what([1,3,2,8,7,2,5,4,0,6,2,15])
#FIXME
```

2. On appelle `what(`L`)` avec L une liste d'entiers naturels.
   (a) Que représente `me` à la fin de la première boucle ?
   (b) Que représente `X` à la fin de la troisième boucle ?
   (c) Que retourne la fonction ?
3. **Bonus :** Quelle est la complexité de cette fonction ?

Annexe : Fonctions et méthodes autorisées

Vous pouvez utiliser les méthodes `append` , `pop` (sans argument uniquement) et la fonction `len` sur les listes :

```
>>> help(list.append)
Help on method_descriptor:    append(...)
    L.append(object) -> None -- append object to end of L

>>> help(len)
Help on built-in function len in module builtins:    len(...)
    len(object)
    Return the number of items of a sequence or collection.

>>> help(list.pop)  # cannot be use with index here...
Help on method_descriptor:
pop(...)
    L.pop() -> item -- remove and return last item.
    Raises IndexError if list is empty.
```

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

Vous pouvez également utiliser la fonction `range` et `raise` pour déclencher les exceptions. Rappels :

```
>>> for i in range(10):
...     print(i, end=' ')
0 1 2 3 4 5 6 7 8 9

>>> for i in range(5, 10):
...     print(i, end=' ')
5 6 7 8 9

>>> raise Exception("blabla")
...
Exception: blabla
```

Vos fonctions

Vous pouvez également écrire vos propres fonctions, dans ce cas elles doivent être documentées (on doit savoir ce qu'elles font).

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