Algorithmique
Partiel n° 1 (P1)

Info-sup S1#
Epita

19 juin 2018 - 9 : 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 (Algorithmes de recherche – *3 points*)

Soit la liste λ suivante :

$$\lambda = \{1, 3, 8, 15, 23, 29, 32, 35, 38, 43, 47, 51, 55\}$$

On cherche la valeur 36 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 2 (Séquences et dichotomie – *3 points*)

1. Parmi les séquences suivantes, lesquelles peuvent correspondre à la suite des étiquettes des valeurs rencontrées lors de la recherche par dichotomie de la valeur 1024 dans une liste triée en ordre croissant ?
   (a) 50, 70, 2048, 75, 1500, 1024
   (b) 50, 75, 2048, 70, 1500, 1024
   (c) 2048, 50, 70, 75, 1500, 1024
   (d) 50, 75, 70, 2048, 1500, 1024
2. Supposons qu'une séquence de recherche soit donnée sous la forme d'une liste. Établir le principe d'une fonction booléenne vérifiant si cette séquence est valide.

Exercice 3 (Liste itérative - *4 points*)

Soit la signature du type algébrique abstrait d'une liste itérative (extrait) présenté ci-dessous.

SORTE
Liste, Place

UTILISE
Entier, Élément

OPÉRATIONS
listevide : → Liste
ième : Liste × Entier → Élément
longueur : Liste → Entier

On se propose d'étendre le type à l'aide de l'opération *mystère* :

OPÉRATIONS
mystère : Liste → Liste

AXIOMES
$\lambda \neq$ liste-vide & $1 \leq k \leq$ longueur(λ) ⇒ ieme(*mystère*(λ),k)=ieme(λ, longueur(λ)-k+1)
longueur(*mystère*(λ)) = longueur(λ)

AVEC
λ : Liste
k : Entier

1. Quel est le nom de l'opération *mystère* ?
2. Implémenter l'opération *mystère* en Python. Attention, cette fonction ne retourne rien : elle doit modifier la liste **en place**.

Exercice 4 (What is it ? – *3 points*)

Soit la fonction `what` ci-dessous :

```
        def what(p, v):
            n = len(p)
            if n < 2:
                raise Exception("not enough")
            (a, b) = p[0]
            (c, d) = p[1]
            i = 1
            while i < n - 1 and b < v:
                i += 1
                (a, b) = (c, d)
                (c, d) = p[i]
            return b + (d - b) * (v - a) / (c - a)
```

1. Donner les résultats des applications suivantes de `what` :
   (a) `what([(0,0), (10,10), (20, 20), (30, 30)], 15)`
   (b) `what([(0,0), (10,20), (20,40), (30,60)], 24)`
   (c) `what([(0,0), (1, 10), (2,100), (3, 1000)], 2.5)`
   (d) `what([(0,3), (1,6), (2,9), (3,10), (4,15)], 20)`
2. Soit L la liste de couples d'entiers `[(`x_0`,`y_0`), (`x_1`,`y_1`), ⋯, (`x_{n-1}`,`y_{n-1}`)]` et Y une valeur numérique. Que calcule `what(`L`, `Y`)` ?

Exercice 5 (Tri par sélection (Select Sort) – *8 points*)

1. Écrire la fonction `minimum(`L`, `d`, `f`)` qui détermine la position de la valeur minimum dans la liste L entre les positions d et f comprises (avec $0 \leq d < f \leq len(L)$).
   Par exemple, dans la liste suivante :

	0	1	2	3	4	5	6	7	8	9
L	4	-5	3	-3	8	-2	0	3	-6	7

   Entre les positions $d = 2$ et $f = 7$, le minimum est à la position 3.
2. Utiliser la fonction précédente pour écrire une fonction qui trie une liste en ordre croissant **en place** (la liste est modifiée par la fonction, aucune autre liste ne doit être utilisée.

   Exemple d'application :

```
>>> L = [4, -5, 3, -3, 8, 2, 0, 3, -6, 7]
>>> selectSort(L)
>>> L
[-6, -5, -3, 0, 2, 3, 3, 4, 7, 8]
```

3. En notant n le nombre d'éléments de la liste, donner pour le tri par sélection :
   (a) le nombre de comparaisons effectuées ;
   (b) le nombre de copies d'éléments.

Annexes

Fonctions et méthodes autorisées

Vous pouvez utiliser la méthode `append` 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.
```

Vous pouvez également utiliser la fonction `range`.

Quelques 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

>>> (a, b) = (1, 2)
>>> (a, b) = (b, a)
>>> (a, b)
(2, 1)
```