Algorithmique
Partiel n° 1 (P1)

Info-sup S1
Epita

4 janvier 2022 - 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 (Colline – *4 points*)

Une liste non vide est une colline si elle est constituée d'une suite (éventuellement vide) strictement croissante suivie d'une suite (éventuellement vide) strictement décroissante.

Les listes suivantes sont des collines :
- $1, 4, 8, 12, 12, 5, 2$
- $12, 25, 40, 52$
- $15, 8, 3$

Les listes suivantes ne sont pas des collines :
- $1, 5, 10, 8, 6, 12$
- $15, 12, 11, 9, 10, 14, 16$

Écrire la fonction `hill(`L`)` qui détermine si la liste L est une colline. Si c'est le cas, elle retourne son point culminant (la valeur la plus haute), -1 sinon. La liste L contient uniquement des entiers positifs ou nuls. Si L est vide la fonction déclenche une exception.

Exemples d'applications :

```
>>> hill([1, 4, 8, 12, 12, 5, 2])
12

>>> hill([12, 25, 40, 52])
52

>>> hill([15, 8, 3])
15

>>> hill([1, 5, 10, 8, 6, 12])
-1

>>> hill([15, 12, 11, 9, 10, 14, 16])
-1

>>> hill([])
Exception: Empty list
```

Exercice 2 (Suppression dans liste triée – *4 points*)

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

Exemples d'applications :

```
>>> L = [12, 8, 5, 4, -3, -7]
>>> delete(L, 5)
True
>>> L
[12, 8, 4, -3, -7]
>>> delete(L, 12)
True
>>> L
[8, 4, -3, -7]
>>> delete(L, 1)
False
>>> L
[8, 4, -3, -7]
```

Exercice 3 (Codage RLE simplifié – *8 points*)

Le but de cet exercice est d'écrire les fonctions de compression/décompression en utilisant une version simplifiée de l'algorithme RLE (Run Length Encoding).

La compression RLE standard permet de compresser des éléments par factorisation, mais uniquement lorsque ceci permet de gagner de la place. Votre algorithme RLE simplifié effectuera cette factorisation dans tous les cas, y compris quand cela prend plus de place que l'objet non compressé.

- Le flux que l'on encode (que l'on veut compresser) est une liste d'éléments, par exemple `['a', 'a, 'a, 'a', 'b', 'b', 'b', 'c']` ;
- le flux encodé (résultat de la compression) est une liste de couples, chaque couple étant composé du nombre (non nul) d'éléments consécutifs identiques, puis de cet élément, par exemple `[(4, 'a'), (3, 'b'), (1, 'c')]`.

D'autres exemples :

- `['b';'b';'b' ; 'c' ; 'a';'a' ; 'b';'b';'b';'b' ; 'a';'a']` est "encodé" `[(3, 'b'); (1, 'c'); (2, 'a'); (4, 'b'); (2, 'a')]`
- `['R'; 'L'; 'E']` est "encodé" `[(1, 'R'); (1, 'L'); (1, 'E')]`
- `['z'; 'z'; 'z'; 'z'; 'z'; 'z']` est "encodé" `[(6, 'z')]`

1. Écrire la fonction `decodeRLE` qui décompresse une liste compressée en RLE.

```
>>> decodeRLE([(6, 'grr')])
['grr', 'grr', 'grr', 'grr', 'grr', 'grr']

>>> decodeRLE([(5, 'a'), (1, 'b'), (3, 'c'), (2, 'd'), (1, 'e')])
['a', 'a', 'a', 'a', 'a', 'b', 'c', 'c', 'c', 'd', 'd', 'e']
```

2. Écrire la fonction `encodeRLE` qui compresse une liste en utilisant l'algorithme RLE.

```
>>> encodeRLE(['grr', 'grr', 'grr', 'grr', 'grr', 'grr'])
[(6, 'grr')]

>>> encodeRLE(['a','a','a','a','a', 'b', 'c','c','c', 'd','d', 'e'])
[(5, 'a'), (1, 'b'), (3, 'c'), (2, 'd'), (1, 'e')]
```

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` et `raise` pour déclencher des exceptions :

```
>>> L = []

>>> for i in range(5):
        L.append(i)

>>> L
[0, 1, 2, 3, 4]

>>> len(L)
5

>>> for i in range(5, 10):
    L.append(i)
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> L.pop()
9

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

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

Liste de couples

Pour créer et récupérer des couples dans une liste, utiliser les syntaxes suivantes :

```
>>> L = [(6, 'grr'), (4, 'z')]
>>> (first, second) = L[0]
>>> first
6
>>> second
'grr'
>>> L.append((5, 'a'))
>>> L
[(6, 'grr'), (4, 'z'), (5, 'a')]
```

Vos fonctions

Vous pouvez également écrire vos propres fonctions, dans ce cas vous devez donnez 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.