S3H - PROG – Examen de fin de module
mercredi 6 janvier 2016 – 10h30 (Vallin)
durée :2h - calculatrices/documents non autorisés (sauf mémento Python 3)

Université de Lille 1 SCIENCES ET TECHNOLOGIES — UFR IEEA Formations en Informatique de Lille 1

[le Barème donné est indicatif, sur 22 points donc 2 points de bonus]

Exercice 1 (7 pt) : TRI RAPIDE et MEDIANE

On donne le code du tri rapide présenté en cours :

```
def my_quicksort(l,u,v):
    if u < v:
        # (a) partition :
        p = l[u]   # pivot
        i = u + 1  # first index
        j = v      # last index
        while i <= j:
            if l[i] < p :
                i += 1
            elif l[j] >= p:
                j -= 1
            else:
                l[i],l[j] = l[j],l[i]
                i += 1
                j -= 1
        # (b) swap «p» and the «last element known to be < p» :
        l[i-1],l[u] = l[u],l[i-1]
        # (c) recursive call x 2 :
        my_quicksort(l,u,i-2)
        my_quicksort(l,i,v)
```

Q1-1 (2.0 pt): Déroulez les différents appels de **`my_quicksort(l,u,v)`** sur **`l=[3,5,2,1,4]`** en choisissant les bonnes valeurs de **`u`** et **`v`** pour que **`l`** soit triée en totalité. Un schéma global des appels est bienvenu ...

Q1-2 (2.0 pt): Expliquez en quoi un tableau déjà *trié* ou *anti-trié* (les éléments étant triés du plus grand vers le plus petit) sera un pire cas pour cet algorithme. Comment y remédier lors du choix du pivot.

On vous propose de modifier ce code pour calculer la médiane ***sans avoir à trier la totalité de l*** *mais* ***uniquement la sous partie*** **utile** ***pour déterminer la médiane.***

On rappelle que la médiane est, après tri théorique, l'élément situé à l'indice ***`len(l)//2`*** *de* ***`l`****.*

Q1-3 (3.0 pt): Proposez un principe de calcul de la médiane, proposez ensuite votre implémentation de cet algorithme dans une fonction **`my_mediane`** que vous paramètrerez et commenterez

---

Exercice 2 (8 pt) : *LISTES en définition récursive (comme en AP2)*

On donne cette définition d'un constructeur de « maillons » pour réaliser une « vrai » liste python :

```
def create (v, n):
   return {'value' : v, 'next' : n}
```

Q2-1 (0.5 pt): Quel est le *type retour* de cette fonction ? Le décrire et *interpréter* également son contenu (*interpréter* en français : pas de code demandé).

Q2-2 (1.0 pt): Déroulez (à l'aide d'un schéma pour chaque ligne) les lignes successives de ce code

```
lc = create ("ping",None)  # (a)
lc = create ("pong",lc)    # (b)
lc['next']['next'] = lc    # (c) {note: ceci n'est pas tres elegant!}
while lc != None:          # (d.1)
   print (lc['value'])     # (d.2)
   lc = lc['next']         # (d.3)
```

Q2-3 (1.5 pt): Donnez le code itératif (ou récursif) de la fonction **`read(ll, p)`** qui retourne la valeur du maillon situé à l'indice **`p (p ≥ 0)`** de la liste **`ll`**. Vous pouvez lever une exception du type **`IndexError`** si la liste est trop courte.

Q2-4 (1.0 pt): Expliquez, en déroulant (au moins) deux exemples (avec de plusieurs schémas pour chaque exemple), le principe de cette fonction :

```
def insert (ll, p, v):
    if p == 0 :
        return create (v,ll)
    else :
        lp = ll
        p -= 1
        while p > 0:
            lp = lp['next']
            p -= 1
        lp['next'] = create (v,lp['next'])
        return ll
```

Q2-5 (1.5 pt): Proposez une version récursive ayant les mêmes types de paramètres et de retour et le même comportement de la fonction de la question précédente.

Q2-6 (2.5 pt): Donnez le code itératif ou récursif de la fonction **`delete(ll, p)`**qui enlève un maillon (mais ne retourne pas sa valeur, cette fonction retournera toujours la liste après effacement du maillon choisi)

Exercice 3 (7 pt) : ***PILES et FILES en utilisant des LISTES en définition récursive***

On donne (à nouveau) cette définition d'un constructeur de « maillons » (« link ») pour réaliser une liste chainée :

```
def create_link (v, n):
   return {'value' : v, 'next' : n}
```

Q3-1 (3.0 pt): Proposez une implémentation d'un module **P**ILE en utilisant comme structure interne une liste de « maillons »,

vous devez disposer des fonctions « push », « pop », « top », « is_empty » et éventuellement de « size »

*On rappelle que contrairement à une PILE où le « dernier » entré est le « premier » sorti (*LIFO*, ou* Last In First Out*), une FILE sort les éléments dans le même ordre dans lequel ils sont rentrés (*FIFO*, ou* First In First Out*)*

Q3-2 (4.0 pt): Proposez une implémentation d'un module **F**ILE en utilisant comme structure interne une liste de « maillons ».

vous devez disposer des fonctions « push/enqueue », « pop/dequeue », « top », « is_empty » et éventuellement de « size »
