IC Devoir 3 - Samedi 25 mars de 11h30 à 12h30

Exercice 1 *Étude d'un algorithme*
Considérons l'algorithme suivant :

Entrées : $a, b \in \mathbb{N}$ avec $b > 0$

$u \leftarrow a$
$v \leftarrow b$
$p \leftarrow 0$
Tant que $v > 0$ faire
  Si v est impair alors $p \leftarrow p + u$ FinSi
  $u \leftarrow u \times 2$
  $v \leftarrow \left\lfloor \dfrac{v}{2} \right\rfloor$
FinTantQue

Sortie : p

1. Appliquer cet algorithme aux nombres $a = 7$ et $b = 23$.
On détaillera, pour chaque étape, le contenu des variables u, v, p dans un tableau :

	u	v	p
Initialisation			
Boucle 1			
Boucle 2			
⋮			

Conjecturer le résultat de cet algorithme dans le cas général.
2. Correction de l'algorithme : montrer que $p + u \times v = ab$ est un invariant de boucle et établir la conjecture précédente.
3. Terminaison de l'algorithme : montrer que le temps d'exécution est fini.
4. Écrire le script d'une fonction PYTHON, `f(a,b)`, réalisant cet algorithme avec les notations usuelles.
5. Proposer une fonction récursive, `f_rec(a,b)`, associé à cet algorithme.

Exercice 2
On considère la fonction `mini(L)` suivante qui retourne la valeur minimale prise par les éléments d'une liste `L` :

```
1 def mini(L):
2     if len(L)==0:
3         return None
4     v=L[0]
5     for i in range(1,len(L)):
6         if L[i]<=v:
7             v=L[i]
8     return v
```

Pour chaque question, il suffira de rappeler sur votre copie les lignes modifiées en précisant leur numéro, voire en introduisant des numéros bis (exemple `1bis`).
1. En modifiant la fonction `mini` ci-dessus, donner une fonction `maxi(L)` qui retourne la valeur maximale prise par les éléments d'une liste `L`.
2. En modifiant la fonction `mini` ci-dessus, donner une fonction `pos_mini(L)` qui retourne la position d'un élément de la liste `L` où la valeur minimale est atteinte.
3. On considère maintenant une liste `t` dont les éléments sont des couples de deux éléments : une chaîne de caractère, un entier.
Exemple : `t=[['Paris,3452],['Mantes',1245], ...]`

En modifiant la fonction `mini` ci-dessus, donner une fonction `mini2(t)` qui retourne un couple dont la valeur numérique associée est minimale.

Exercice 3
Soit un entier naturel n non nul et une liste `t` de longueur n dont les termes valent 0 ou 1. Le but de cet exercice est de trouver le nombre maximal de 0 contigus dans `t` (c'est-à-dire figurant dans des cases consécutives). Par exemple, le nombre maximal de zéros contigus de la liste `t1` suivante vaut 4 :

i	0	1	2	3	4	5	6	7
t1[i]	0	1	1	1	0	0	0	1

i	8	9	10	11	12	13	14
t1[i]	0	1	1	0	0	0	0

1. Écrire une fonction `nombreZeros(t,i)`, prenant en paramètres une liste `t`, de longueur n, et un indice i compris entre 0 et $n-1$.

- Si `t[i]` vaut 1 alors la fonction revoie 0.
- Sinon, elle renvoie le nombre de zéros consécutifs dans `t` à partir de `t[i]` inclus.

Par exemple, les appels `nombreZeros(t1,4)`, `nombreZeros(t1,1)` et `nombreZeros(t1,8)` renvoient respectivement les valeurs 3, 0 et 1.
2. Comment obtenir le nombre maximal de zéros contigus d'une liste `t` connaissant la liste des `nombreZeros(t,i)` pour $0 \leq i \leq n-1$?
En déduire une fonction `nombreZerosMax(t)`, de paramètre `t`, renvoyant le nombre maximal de 0 contigus d'une liste `t` non vide. On utilisera la fonction `nombreZeros`.
3. Quelle est la complexité de la fonction `nombreZerosMax` construite à la question précédente ? *(cas le meilleur et cas le pire)*
4. Trouver un moyen simple, toujours en utilisant la fonction `nombreZeros`, d'obtenir un algorithme plus performant et estimer la nouvelle complexité.

Exercice 4 Réécrire avec une boucle `while` l'instruction suivante :

```
u=2
for i in range(1,n):
    u=u+1/i
```