Informatique en CPGE (2018-2019)
Exercices : récursivité

Exercice 1

Considérons la fonction définie ci-dessous qui prend en paramètre un entier naturel n.

```
def f(n):
    nstr=str(n)
    s=0
    for c in nstr:
        s=s+int(c)
    return s
```

1. Que renvoie l'appel f(52431) ?
2. Ecrire une fonction g, version récursive de la fonction f.
3. Ecrire une fonction récursive h qui prend en paramètre une chaîne de caractères c et renvoie la chaîne obtenue en inversant l'ordre des caractères. Par exemple f('abc') renvoie la chaîne 'cba'.
4. En déduire une fonction **renverse** qui prend en paramètre un entier naturel n et renvoie l'entier naturel obtenu en inversant l'ordre des chiffres dans l'écriture de n. Par exemple renverse(5042) renvoie 2405 ; attention renverse(370) renvoie 73.
   La fonction **renverse** utilise la fonction précédente h.

Exercice 2

On considère une fonction récursive qui additionne deux entiers naturels en ajoutant simplement une unité à chaque appel récursif.

```
def add(a,b):
    if b==0:
        return a
    else:
        return add(a,b-1)+1
```

1. Déterminer en fonction de b le nombre d'appels de la fonction **add** pour effectuer a + b.
2. Expliquer pourquoi la terminaison n'est assurée que dans le cas $b \in \mathbb{N}$.
3. Démontrer la correction de la fonction, autrement dit, prouver que l'appel **add(a, b)** renvoie bien la somme a + b.
4. Ecrire en suivant le même modèle une fonction récursive **mul** qui effectue la multiplication de deux nombres entiers naturels. Utiliser la fonction **add** ; l'opérateur " + " n'est pas autorisé.
5. Ecrire une fonction récursive calculant a^b, avec des valeurs de b entières positives ou nulles. Utiliser la fonction **mul** ; ici l'opérateur " * " n'est pas autorisé.

Exercice 3

A partir de l'observation $x^{2k} = (x^k)^2$ et $x^{2k+1} = x(x^k)^2$, écrire une fonction récursive **puissance** prenant en paramètres un nombre x quelconque et un entier naturel n et renvoyant la valeur x^n.

Exercice 4

Considérons les deux fonctions **compte1** et **compte2** définie ci-dessous.

```
def compte1(n):
    if n>=0:
        print(n)
        compte2(n-1)

def compte2(n):
    if n>=0:
        compte1(n-1)
        print(n)
```

Le paramètre n est un entier naturel. Déterminer sans écrire de code sur machine le résultat de l'appel compte1(3) puis celui de l'appel compte2(3).

Exercice 5

Soit f définie sur $\mathbb{R}$ par $f(x) = \dfrac{x^2 + \cos x}{2}$ et (u_n) la suite définie par $u_0 = 2$ et $u_{n+1} = f(u_n)$ pour tout $n > 0$.

La solution α sur l'intervalle $[0; 2]$, de l'équation $f(x) = x$, autrement dit le point fixe, est donnée par la limite en $+\infty$ de la suite (u_n).

Ecrire la définition de la fonction f puis la définition de la suite (u_n) en version récursive.

Afficher les vingt premiers termes de la suite et estimer alors une valeur approchée de α à 10^{-4} près.

Exercice 6

Ecrire une fonction récursive qui résout par dichotomie une équation du type $f(x) = 0$ sur un intervalle $[a; b]$ tel que $f(a)$ et $f(b)$ sont de signes contraires.

Utiliser ce programme pour déterminer une solution approchée à 10^{-4} près de l'équation de l'exercice précédent $\dfrac{x^2 + \cos x}{2} = x$ sur l'intervalle $[0; 2]$.

Déterminer une solution approchée à 10^{-4} près de cette même équation sur l'intervalle $[2; 4]$.

Exercice 7

Un palindrome est un mot qui peut se lire de gauche à droite ou de droite à gauche. Un nombre palindrome est un nombre qui peut se lire de gauche à droite ou de droite à gauche. Par exemple les mots "non", "radar", "ressasser" ou les nombres 33, 272, 313 sont des palindromes.

1. La fonction **palindrome** présentée ci-dessous prend en paramètre une chaîne de caractères et renvoie True si la chaîne est un palindrome et False sinon.

Noter que le paramètre peut être aussi du type **list**.

Ecrire une version récursive de cette fonction.

```
# version non récursive
def palindrome(c):
    "c est de type str ou list"
    test=True
    while len(c)>1:
        if c[0]!=c[-1]:
            return False
        else:
            c=c[1:-1]
    return test
```

2. Ecrire une fonction **premier** basique qui prend en paramètre un entier naturel n et renvoie True si le nombre n est premier et False sinon.

Déterminer alors la liste des nombres premiers palindromes inférieurs à 1000.

Déterminer le premier nombre premier palindrome supérieur à 1000.