Informatique PCSI
TP 5 suite : algorithmes récursifs

1 Énoncé des exercices

1.1 Exercice 1

La valeur du pgcd (plus grand commun diviseur) de deux entiers naturels a et b est calculable à l'aide de l'algorithme d'Euclide : $\text{pgcd}(a,b) = \text{pgcd}(b,r)$ où r est le reste dans la division euclidienne de a par b. En voici une version itérative :

```
def pgcd(a, b):
    if b == 0:
        return a
    while b != 0:
        a, b = b, a % b
    return a
```

Écrire une version récursive de cette fonction.

1.2 Exercice 2

Une fonction `nettoie` prend en paramètre une liste triée et élimine les éléments identiques. Si la liste est `liste=[1, 1, 2, 6, 6, 6, 8, 8, 9]`, alors cette liste a pour valeur `[1, 2, 6, 8, 9]` après l'instruction `nettoie(liste)`. Écrire une version récursive.

```
def nettoie(L):
    n = len(L)
    k = 0
    while k < n - 1:
        if L[k] != L[k+1]:
            k = k + 1
        else:
            del L[k] # instruction del (delete): supprime l'élément L[k]
            n = len(L)
```

La fonction récursive prend deux paramètres, la liste et un indice `k` qui a pour valeur initiale par défaut 0.

1.3 Exercice 3

Utilisation du module Turtle
Tester le programme ci-dessous puis en écrire une version récursive.

```
from turtle import *

couleurs = ['blue', 'green', 'yellow', 'orange', 'red', 'purple']
bgcolor('black')

def dessin():
    for i in range(180):
        color(couleurs[i%6])
        forward(i)
        right(59)

dessin()
```

1.4 Exercice 4

On considère la fonction récursive `dicho_rec` définie dans le document préparatoire.

1. On définit une liste par : `liste = [2, 5, 7, 8, 10, 13, 15, 16, 17, 20, 23, 24]`.
   On écrit `dicho_rec(10, liste, 0, len(liste)-1)`.
   Détailler l'exécution de la fonction en précisant les valeurs prises successivement par les trois variables `g`, `d` et `m`.
2. On conserve la même liste et on écrit `dicho_rec(18, liste, 0, len(liste)-1)`. Expliquer avec précision l'exécution de la fonction.

1.5 Exercice 5

La fonction `palindrome` est écrite ci-dessous de manière itérative avec une boucle `for` et une interruption de boucle. Écrire une version récursive.

```
def palindrome(ch):
    n = len(ch)
    for i in range(n//2):
        if ch[i] != ch[n-1-i]:
            return False
    return True
```