Cours 2 : Corrigé des exercices

Exercices 1, 2 & 4

Exemple 1. *Division euclidienne*

```
def division_euclidienne(a, b) :
    '''renvoie le quotient et le reste de la division euclidienne de a par b'''
    q = 0
    r = a
    while r >= b :
        q = q + 1
        r = r - b
    return (q, r)
```

— La précondition est $a, b \in \mathbb{N} \times \mathbb{N}^\star$.
— La postcondition est $a = b \cdot q + r$ avec $q, r \in \mathbb{N} \times \mathbb{N}$.
— Un variant de boucle est r.
— Un invariant de boucle est P_k : "$a = q_k \cdot b + r_k$".

Exemple 2. *Plus Grand Commun Diviseur*

```
def PGCD(a, b) :
    '''Recherche du PGCD de deux entiers a et b'''
    u = a
    v = b
    while u != v :
        if u > v :
            u = u - v
        else :
            v = v - u
    return (u)
```

Remarque 1. *pgcd(a,b) est noté* $a \wedge b$.

— La précondition est $a, b \in \mathbb{N}^\star \times \mathbb{N}^\star$.
— La postcondition est $u = a \wedge b$.
— Un variant de boucle est $u + v$ ou $\max(u, v)$.
— Un invariant de boucle est P_k : "$u_k \wedge v_k = a \wedge b$".

Exemple 3.

```
# Un programme a tester
c = 0
while p > 0 :
    if c == 0 :
        p = p - 2
        c = 1
    else :
        p = p + 1
        c = 0
```

— La précondition est $p \in \mathbb{N}^\star$.
— La postcondition est $p = 0$.
— Un variant de boucle est $2 \cdot (p + c) + c = 2 \cdot p + 3 \cdot c$.

Exercice 3

```
# Suite de Syracuse
def Syracuse(n) :
    '''fonction affichant les termes de la suite de Syracuse
    d'un entier n strictement positif.'''
    Sn = [n] # nombre de depart
    while Sn[-1] != 1 :
        # tant que le terme de la suite est different de 1
        # la terminaison est une conjecture qui, en depit de la simplicite de son enonce, \
        # n'a toujours pas ete demontree ni infirmee...
        if Sn[-1] % 2 : Sn.append(3 * Sn[-1] + 1)
        # si sn est impair, on le remplace par 3 * sn + 1
        else : Sn.append(Sn[-1] // 2)
        # sinon, on le divise par 2
    return (Sn)
```

Exercice 5

```
# Recherche de la premiere occurrence d'un element dans une liste
def occurrence(elt, L) :
    '''fonction qui recherche l'indice de la premiere occurrence
    de l'element elt dans une liste L non triee
    et qui renvoie None sinon.'''
    i = 0   # initialisation de l'indice recherche
    while L[i] != elt and i < len(L) :
        # variant de boucle : len(L) - i
        # invariant de boucle : elt n'est pas dans les i premiers elements de
        # la liste
        i = i + 1
    if i == len(L) -1 :
        return(None)
    else :
        return(i)
```

Exercice 6

```
# Calcul des termes d'une suite definie par recurrence
def Un(n) :
    '''Fonction qui permet de calculer de maniere recursive le nieme terme
    de la suite Un telle que :
    U0 = 2
    Un = 1/2 * ( Un-1 + 3 / Un-1 )'''
    if n == 0 : return (2)  # cas de base
    else :
        return( 0.5 * (Un(n - 1) + 3 / Un(n - 1)) )
```

Exercice 7

```
# Conversion entier -> binaire

def conversion_it(n) :
    '''renvoie la conversion de l'entier n en binaire sour la forme d'une
    chaine de caracteres.'''
    res = ''    # initialisation de la chaine de caractere resultat
    while n != 0 :
        res = str(n % 2) + res # On concatene le reste trouve a gauche
        n = n // 2
    return(res)

def conversion_rec(n) :
    '''renvoie la conversion de l'entier n en binaire sour la forme d'une
    chaine de caracteres selon un algorithme recursif.'''
    if n == 1 or n == 0 :
        # cas de base
        return(str(n))
    else :
        return(conversion_rec(n // 2) + str(n % 2))
```