Correction_Fonctions_recursives 1

Exercice 1

Soit une chaine de caractères, écrire un algorithme récursif permettant de déterminer sa longueur

```
def longueur(ch):
    if not ch:
        return 0
    else:
        return 1+longueur(ch[1:])

ch = "Take It Easy"

print(longueur(ch))
```

Exercice 2

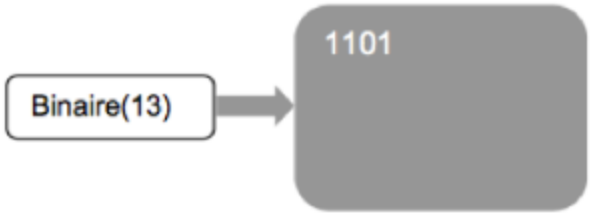
Rendre récursive la fonction somme suivante :

```
def somme(L):
    s=0 :
    for val in L :
        s+=val
    return s
```

```
def somme(L):
    if not L:
        return 0
    return L[0]+somme(L[1:])
```

Exercice 3

Pour convertir un nombre entier positif N de la base décimale à la base binaire, il faut opérer par des divisions successives du nombre N par 2. Les restes des divisions constituent la représentation binaire.




```python
def binaire(N):
    if N == 0:
        return []
    return binaire(N//2)+[N % 2]

print(binaire(13))
```

Exercice 4

Un nombre N est pair si (N−1) est impair, et un nombre N est impair si (N−1) est pair.
Ecrire deux fonctions récursives mutuelles pair(N) et impair(N) permettant de savoir si un nombre N est pair et si un nombre N est impair.

```python
def Pair(N):
    if N == 1:
        return False
    return Impair(N-1)

def Impair(N):
    if N == 1:
        return True
    return Pair(N-1)
```

Exercice 5

Soit un tableau X de N entiers, écrire une fonction récursive simple permettant de déterminer le maximum du tableau

```
def maximum(T):
    if len(T) == 1:
        return T[0]

    # principe de la recherche dichotomique
    m = len(T)//2
    max1 = maximum(T[:m])
    max2 = maximum(T[m:])

    if max1 > max2:
        return max1
    return max2
```

Exercice 6

Un tableau X est trié par ordre croissant si $x(i) \leq x(i+1), \forall i$, écrire un algorithme récursif permettant de vérifier qu'un tableau X est trié ou non

```
def Esttrier(T):
    if len(T) == 0 or len(T) == 1:
        return True
    if T[0] <= T[1]:
        return Esttrier(T[1:])
    return False

T = [1, 2, 3, 4, 4, 5, 7, 8]
print(Esttrier(T))
```

Exercice 7

Un mot est un palindrome si on peut le lire dans les deux sans de gauche à droite et de droite à gauche. Exemple KAYAK est un palindrome. Ecrire une fonction récursive permettant de vérifier si un mot est palindrome.

```
def palindrome(ch):
    if len(ch) == 1 or len(ch)==0:
        return True
    if ch[0] == ch[-1]:
        return palindrome(ch[1:len(ch)-1])
    return False

ch = "KAYAK"
print(palindrome(ch))
```

Exercice 8

Soit un tableau d''entiers contenant des valeurs 0 ou bien 1. On appel composante connexe une suite contigue de nombres égaux à 1. On voudrait changer la valeur de chaque composante connexe de telle sorte que la première composante ai la valeur 2 la deuxième ai la valeur 3, la 3ème ait la valeur 4 et ainsi de suite. Réaliser deux fonctions :

1. La première fonction n'est pas récursive et a pour rôle de chercher la position d'un 1 dans un tableau.
2. La deuxième fonction est récursive. Elle reçoit la position d'un 1 dans une séquence et propage une valeur x à toutes les valeur 1 de la composante connexe.

```
# Trouver l'indice du premier 1
def trouver(T):
    for i in range(len(T)):
        if T[i] == 1:
            return i
    return None
```

```python
# Fonction principale
def propage(T, x, val):
    i = trouver(T)
    if i is None:
        return

    T[i] = val
    if (len(T)-i+1) > 2:
        if T[i+1] == 1:
            propage(T, x, val+1)
        else:
            propage(T, x, x)



# tester la fonction
T = [0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1]
x = 2
propage(T, x, x)

print(T)
```

Exercice 9

Soit une image binaire représentés dans une matrice à 2 dimension. Les éléments m[i][j] sont dits pixels et sont égaux soit à 0 soit à 1. Chaque groupement de pixels égaux à 1 et connectés entre eux forment une composante connexe(figure). L'objectifs est de donner une valeur différente de 1 à chaque composante (2 puis 3 puis 4 etc.)

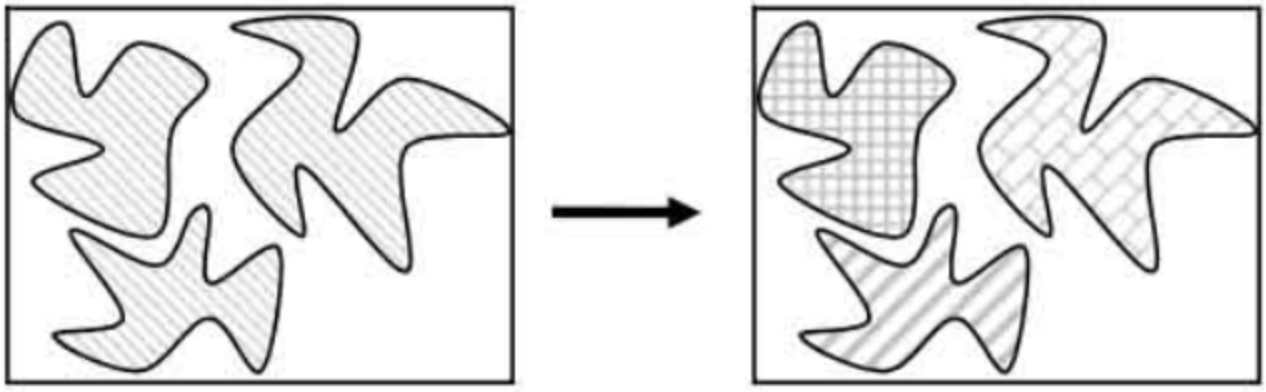


Image binaire

Image étiquetée

1. Ecrire une fonction récursive propager permettant de partir d'un point (i,j) situé à l'intérieur d'une composante connexe et de propager une étiquette T à tous les pixels situés à l'intérieur de la composante.
2. Ecrire une fonction etiqueter permettant d'affecter une étiquette différente à chaque composante connexe.

```
def propager(M, i, j, val):
    if M[i][j] == 0:
        return
    M[i][j] = val
    if((i-1) >= 0 and M[i-1][j] == 1):  # l'élément en haut
        propager(M, i-1, j, val)
    if((i+1) < len(M) and M[i+1][j] == 1):  # l'élément en bas
        propager(M, i+1, j, val)
    if((j-1) < len(M) and M[i][j-1] == 1):  # l'élément a
gauche
        propager(M, i, j-1, val)
    if((j+1) < len(M) and M[i][j+1] == 1):  # l'élément a
droite
        propager(M, i, j+1, val)

def etiqueter(M):
    L, C = len(M), len(M[0])
    val = 2
```

```
    for i in range(L):
        for j in range(C):
            if(M[i][j] == 1):
                propager(M, i, j, val)
                val += 1

M = [[0, 0, 1, 0], [0, 1, 0, 1], [1, 1, 1, 0], [0, 1, 1, 0]]
print(M)
etiqueter(M)
print(M)
```