***Algorithmique et programmation avancée***
***« Récursivité »***

***Exercice N° 2 :***

```
def somme(n) :
   if n==0 :
      return 0
   return n+ somme(n-1)
```

***Exercice N° 3 :***

```
def puissance(x, n):
   if n==0 :
      return 1
   return x*puissance(x, n-1)
```

***Exercice N° 4 :***

```
def somme(L):
   if len(L)==1 :
      return L[0]
   return L[0] + somme(L[1:])
```

***Exercice N° 5 :***

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

***Exercice N° 6 :***

```
def fib(n):
   if n==0 or n==1 :
      return n
   return fib(n-2) + fib(n-1)
```

***Exercice N° 7 :***

```
def chiffre(n, k):
   if k==1 :
      return n%10
   return chiffre(n//10, k-1)
```

***Exercice N° 8 :***

```
def combinaisons(n, p):
   if p==0 or p==n :
      return 1
```

```
    return combinaisons(n-1, p) + combinaisons(n-1, p-1)
```

***Exercice N° 8 :***

```
def bin(n):
    if n==0 :
        return 0
    return bin(n//2)*10 + n%2
```

***Exercice N° 9 :***

```
def taille(L) :
    if L==[]:
        return 0
    return 1 + taille(L[1:])
```

***Exercice N° 10 :***

```
def rechercheDichoRec(L, v, i, j):
    m=(i+j)//2
    if L[m]==v :
        return m
    if i>j :
        return -1
    if v < L[m]
        return rechercheDichoRec(L, v, i, m-1)
    return rechercheDichoRec(L, v, m+1, j)
```

***Exercice N° 11 :***

```
def permutations(string):
    permutation_list = []
    if len(string) == 1:
        return [string]
    else:
        for char in string:
            for a in permutations(string.replace(char, "", 1)):
                permutation_list.append(char + a)
    return permutation_list
```

***Exercice N° 12 :***

```
def R(n, i):
    if i>n:
        return 0
    return sqrt(i+R(n,i+1))
```