Exercice 1 *3 points*

1. Si on donne un nombre impair comme argument à cette fonction récursive `pair()` (par exemple `pair(7)`), alors l'appel récursif sera infini et la fonction ne renverra jamais de valeur.

2.

```
def pair(n):
    if n == 0 :
        return True
    elif  n == 1 :
        return False
    else :
        return pair(n-2)
```

Exercice 2 *3 points*

1. `myst([1,5,2,10,3],3)` = `lst[3] + myst([1,5,2,10,3],2)`
   = `10 + lst[2] + myst([1,5,2,10,3],1)`
   = `10 + 2 + lst[1] + myst([1,5,2,10,3],0)`
   = `10 + 2 + 5 + lst[0] + myst([1,5,2,10,3],-1)`
   = `10 + 2 + 5 + 1 + 0`
   = `18`
2. Il faudrait vérifier que `k` un nombre entier compris entre `0` et `len(lst)-1`.

Exercice 3 *3 points*

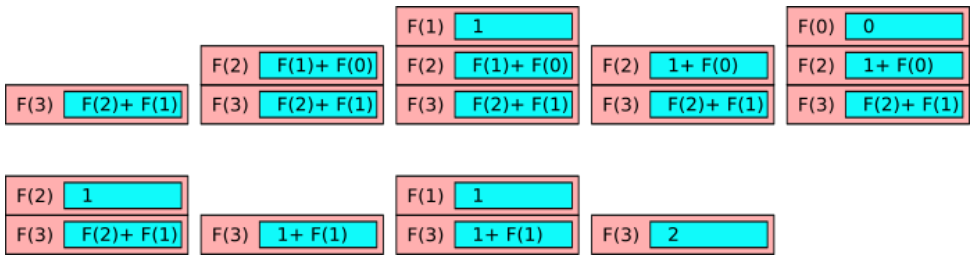
```
def pgcd(a,b):
    if b == 0 :
        return a
    else :
        return pgcd(b, a % b)
```

Exercice 4 *5 points*

1.

```
def fibo(n):
    if n == 0 :
        return 0
    elif n == 1 :
        return 1
    else :
        return fibo(n-1) + fibo(n-2)
```

2.



Exercice 5 *6 points*

1. a. $Ack(0,1) = 1+1 = 2$
   b. $Ack(0,2) = 2+1 = 3$
   c. $Ack(1,0) = Ack(0,1) = 2$
   d. $Ack(2,0) = Ack(1,1) = Ack(0, Ack(1,0)) = Ack(0,2) = 3$
   e. $Ack(1,2) = Ack(0, Ack(1,1)) = Ack(0,3) = 4$
2.

```
def ack(m, n):
    if m == 0 :
        return n + 1
    if m > 0 and n == 0 :
        return ack(m-1, 1)
    if  m > 0  and n > 0 :
        return ack(m-1, ack(m, n-1))
```