Exercice 1 (2 points) *Démontrez que le code suivant de division entière est valide*

```
def q(a,b):
        q=0
        while a - q*b >= b:
                q=q+1
        return q
```

Correction. Le code est correct si la valeur q renvoyée est l'unique entier tel que $a = qb + r$, où $0 \leq r < b$ est un entier.

Si $a < b$, on ne rentre pas dans la boucle et on retourne $q = 0$; correct. Si $a \geq b$, on sort de la boucle avec

1. $a - qb < b$ et
2. $a - (q-1)b \geq b$

or (1) implique $r = a - qb < b$, et (2) implique $r = a - qb \geq 0$; correct.

Exercice 2 ($5 \times 0.5 = 2.5$ points) *Mettez sous la forme $\Theta(n^\alpha \log^\beta n)$ les expressions suivantes*

1. $\frac{\log_b n}{\log_a n}$
2. $a^{\log_b n}$
3. $\sum_{i=1}^{i=n} \frac{1}{i}$
4. $\log(n!)$
5. $\sum_{i=1}^{n} i^c$

où $a, b, c \geq 2$ sont des entiers constants.

Correction.

1. $\frac{\log_b n}{\log_a n} = \Theta(1)$
2. $a^{\log_b n} = \Theta(n^{\log_b a})$
3. $\sum_{i=1}^{i=n} \frac{1}{i} = \Theta(\log n)$
4. $\log(n!) = \Theta(n \log n)$
5. $\sum_{i=1}^{n} i^c = \Theta(n^{c+1})$

Exercice 3 (2 points) *Démontrez en détail votre réponse à la dernière expression de l'exercice précédent.*

Correction. Soit $f(n) = \sum_{i=1}^{n} i^c$. Déjà $f(n) \leq \sum_{i=1}^{n} n^c = n^{c+1}$ d'où $f(n) = O(n^{c+1})$. De plus,

$$f(n) \geq \sum_{i=\lceil \frac{n}{2} \rceil}^{n} \left(\frac{n}{2}\right)^c = 2^{-c} \sum_{i=\lceil \frac{n}{2} \rceil}^{n} n^c \geq 2^{-c} \left\lfloor \frac{n}{2} \right\rfloor n^c$$

on a, par exemple pour $n \geq 2$, $\lfloor \frac{n}{2} \rfloor \geq \frac{n}{4}$, d'où $f(n) \geq 2^{-c-2} n^{c+1} = \Omega(n^{c+1})$.

Exercice 4 ($0.5 + 1 + 2 = 3.5$ points) *On considère le code suivant:*

```
A=[0 for i in range(n+1)]
A[1]=1
def ft(i):
        global A,n
        if i==0 or A[i]>0:
                return A[i]
        else:
                A[i]=ft(i-1)+ft(i-2)
                return A[i]
```

1. *Quelle est la valeur retournée par l'appel à* `ft(12)` *?*
2. *Quelle est la complexité de* `ft(n)` *?*
3. *Démontrer votre réponse au 2.*

Correction.

1. 144
2. $\Theta(n)$
3. L'appel à `ft(n)` engendre récursivement les appels à `ft(n-1)`, `ft(n-2)`,..,`ft(0)` donc la complexité de `ft(n)` est $\Omega(n)$. Il ne peut y avoir plus de 2 appels récursifs à `ft(i)` puisque le premier appel donne une valeur > 0 à `A[i]` pour $i \geq 2$; donc `ft(n)` est $O(n)$.

Exercice 5 (1+1+1 = 3 points) *On a effacé des parties du code python suivant:*

```
def fc(n):
        if n==0:
                return [0,1]
        else:
                        =
                return [v,u+v]
```

1. *Complétez le code de* `fc(n)` *pour que la valeur retournée soit* $\frac{1}{\sqrt{5}}(\phi^n - \bar{\phi}^n)$*, avec* $\phi = \frac{1+\sqrt{5}}{2}$ *et* $\bar{\phi} = \frac{1-\sqrt{5}}{2}$*.*
2. *Quelle est la complexité de* `fc(n)` *?*
3. *Démontrer votre réponse au 2.*

Correction.

1.
```
def fc(n):
        if n==0:
                return [0,1]
        else:
                [u,v] = fc(n-1)
                return [v,u+v]
```

2. $\Theta(n)$
3. L'appel à `fc(n)` engendre récursivement les appels à `fc(n-1)`, `fc(n-2)`,..,`fc(0)` donc la complexité de `fc(n)` est $\Omega(n)$. Il n'y a qu'un seul appel récursif; donc `fc(n)` est $O(n)$.

Exercice 6 (0.5 + 2 = 2.5 points) *On considère l'algorithme d'Euclide renvoyant $pgcd(a, b)$.*

```
def e(a,b):
        if a*b==0:
                return a+b
        else:
                return e(b,a%b)
```

1. *Quelle est la complexité de* `e(a,b)` *?*
2. *Démontrez votre réponse en 1.*

Correction.

1. La complexité est $\Theta(n)$ où $n = \log(\max\{a, b\})$ représente la taille des données
2. La complexité est au minimum de l'ordre de la taille $\Omega(n)$ des données. Soit

$$r_i = \begin{cases} \max\{a, b\} & \text{si } i = 0 \\ \min\{a, b\} & \text{si } i = 1 \\ \text{l'unique entier } r_i \text{ tel qu'il existe un entier } q_i \text{ tel que } r_{i-2} = q_i r_{i-1} + r_i & \text{si } i \geq 2 \end{cases}$$

L'appel initial $e(a, b)$ est équivalent à $e(r_0, r_1)$ si $a \geq b$; sinon $a = 0.b + a$, et le premier appel récursif est équivalent à $e(r_0, r_1)$. Les appels récursifs suivants sont $e(r_1, r_2)$, $e(r_2, r_3)$, ..., $e(r_n, 0)$ avec $r_{i-2} = q_i r_{i-1} + r_i$ où $q_i \geq 1$ pour $i = 2, \ldots, n + 1$ ($r_{n+1} = 0$). Donc r_0 est supérieur ou égal au $n$ème terme de la suite de Fibonacci soit $r_0 = \Omega(\phi^n)$ avec $\phi > 1$. D'où $n = O(\log r_0)$.

Exercice 7 (1+1+1+1 = 4 points) *Considérons un algorithme dont la taille des entrées est représentée par un entier n, et dont le temps d'exécution $T(n)$ satisfait:*

$$T(n) = \begin{cases} \Theta(1) & \text{si } n < b \\ aT(n - b) + \Theta(n^c) & \text{si } n \geq b \end{cases}$$

pour des entiers $a, b, c \geq 1$.

1. *Démontrez que $T(n) = \Theta\Big(a^q + \sum_{i=0}^{q-1} a^i (n - ib)^c\Big)$ où q est le quotient de la division entière de n par b.*
2. *Démontrez que si $a \geq 2$, l'algorithme est exponentiel.*
3. *Démontrez que $\sum_{i=0}^{q-1} (n - ib)^c = \sum_{i=1}^{q} (r + ib)^c$*
4. *Démontrez que si $a = 1$, alors $T(n) = \Theta(n^{c+1})$.*

Correction.

1. Étant donné un entier n, on note $n = qb + r$ avec $q, r \in \mathbb{N}$ et $r < b$. Soit $f(n) = a^q + \sum_{i=0}^{q-1} a^i (n - ib)^c$. Soient α', β' les constantes de la fonction $\Theta(n^c)$, et α'', β'' les constantes de la fonction $\Theta(1)$. On définit $\alpha = \min\{1, \alpha', \alpha''\}$ et $\beta = \max\{1, \beta', \beta''\}$, ainsi on a

$$\alpha \leq f(0.b + r) = a^0 = 1 \leq \beta$$

d'où

$$\alpha f(0.b + r) \leq T(0.b + r) \leq \beta f(0.b + r)$$

Par ailleurs si

$$\alpha f(qb + r) \leq T(qb + r) \leq \beta f(qb + r)$$

alors

$$\alpha f((q + 1)b + r) \leq T((q + 1)b + r) \leq \beta(f((q + 1)b + r)$$

puisque, avec $n = (q + 1)b + r$, on a:

$$\begin{array}{rcccl} aT(qb + r) + \alpha n^c & \leq & T(n) & \leq & aT(qb + r) + \beta n^c \\ a \times \alpha\Big(a^q + \sum_{i=0}^{q-1} a^i (n - b - ib)^c\Big) + \alpha n^c & \leq & T(n) & \leq & a \times \beta\Big(a^q + \sum_{i=0}^{q-1} a^i (n - b - ib)^c\Big) + \beta n^c \\ \alpha\Big(a^{q+1} + \sum_{i=0}^{q-1} a^{i+1} (n - b - ib)^c\Big) + \alpha n^c & \leq & T(n) & \leq & \beta\Big(a^{q+1} + \sum_{i=0}^{q-1} a^{i+1} (n - b - ib)^c\Big) + \beta n^c \\ \alpha\Big(a^{q+1} + \sum_{i=1}^{q} a^i (n - ib)^c\Big) + \alpha n^c & \leq & T(n) & \leq & \beta\Big(a^{q+1} + \sum_{i=1}^{q} a^i (n - ib)^c\Big) + \beta n^c \\ \alpha f((q + 1)b + r) = \alpha\Big(a^{q+1} + \sum_{i=0}^{q} a^i (n - ib)^c\Big) & \leq & T(n) & \leq & \beta\Big(a^{q+1} + \sum_{i=0}^{q} a^i (n - ib)^c\Big) = \beta f((q + 1)b + r) \end{array}$$

2. Puisque $b > r = n - bq$, on a $1 > \frac{n}{b} - q$ d'où $a^q \geq a^{\frac{n}{b}-1} = \frac{1}{a}a^{n/b}$, alors $T(n) = \Omega(d^n)$ avec $d = a^{1/b}$. Ainsi, puisque $d^b = a$ avec $b \geq 1$, si $a > 1$, alors $d > 1$ et l'algorithme est exponentiel.

3. $\sum_{i=0}^{q-1}(n-ib)^c = \sum_{i=0}^{q-1}(qb+r-ib)^c = \sum_{i=0}^{q-1}(r+(q-i)b)^c = \sum_{i=1}^{q}(r+ib)^c$ où la dernière égalité provient du fait que $\{q-0, q-1, \ldots, q-(q-1)\} = \{1, 2, \ldots, q\}$.

4. Il suffit de montrer que $\sum_{i=1}^{q}(r+ib)^c = \Theta(n^{c+1})$ ce qui découle de

$$b^c \sum_{i=1}^{q} i^c = \sum_{i=1}^{q}(ib)^c \leq \sum_{i=1}^{q}(r+ib)^c \leq 2\sum_{i=1}^{q}(ib)^c = 2b^c \sum_{i=1}^{q} i^c$$

Exercice 8 (2 points) *Le théorème de Bachet-Bezout énonce que pour toute paire d'entiers $a, b \in \mathbb{N}$, il existe deux relatifs $u, v \in \mathbb{Z}$ tels que $au + bv = pcgd(a, b)$.*

Utilisez le code suivant pour démontrer qu'il existe une infinité de couples u, v.

```
def B(a,b):
        """on suppose a>=b"""
        if b==0:
                return [1,0,a]
        else:
                [x,y,d]=B(b,a%b)
                return [y,x-(a//b)*y,d]
```

Correction. L'appel à `infty(a,b)` donne une infinité de couples u, v

```
n=0
def B(a,b):
        global n
        if b==0:
                return [1,n,a]
        else:
                [x,y,d]=B(b,a%b)
                return [y,x-(a//b)*y,d]
def infty(a,b):
        while(True):
                B(a,b)
                n=n+1
```