

Tout document, calculatrice, téléphone portable ou ordinateur est interdit.

Les exercices sont indépendants. Si l'on ne sait pas justifier une question, on peut néanmoins utiliser la réponse dans la suite.

**Durée : 1 heure**

► **Exercice 1. (Échelles de comparaisons)**

Dans cet exercice  $f, g, h$  sont des fonctions positives.

1. Montrer que si  $f \in O(g)$  et  $g \in O(h)$  alors  $f \in O(h)$ .



.....  
C'est une question qui a été traitée en cours :

- Par définition  $f \in O(g)$  signifie qu'il existe un entier  $n_0$  et une constante  $k > 0$  tel que pour tout  $n > n_0$ , on a

$$f(n) \leq k \cdot g(n).$$

- De même  $g \in O(h)$  signifie qu'il existe un entier  $n_1$  et une constante  $l > 0$  tel que pour tout  $n > n_1$ , on a


$$g(n) \leq l \cdot h(n).$$

- On en déduit que si  $n > \max(n_0, n_1)$  on a

$$f(n) \leq k \cdot g(n) \leq k \cdot l \cdot h(n).$$

- Donc pour  $N = \max(n_0, n_1)$  et  $C = k \cdot l$ , on a si  $n > N$ ,

$$f(n) \leq C \cdot h(n).$$

..... 

On considère les fonctions suivantes :

$$f_1 = 17x, \quad f_2 = 5x^3 + 8x + 2, \quad f_3 = 2 \log(x) + 1, \quad f_4 = 3x + 1, \quad f_5 = 2^x + x$$

2. Remplir un tableau  $5 \times 5$  en mettant dans la case à l'intersection de la ligne  $i$  et de la colonne  $j$  le symbole qui convient selon la règle :

- $\Theta$  si  $f_i \in \Theta(f_j)$ ,
- $O$  si  $f_i \in O(f_j)$  mais  $f_i \notin \Theta(f_j)$ ,
- $\times$  sinon.



On rappelle qu'un polynôme de degré  $d$  se comporte  $x^d$ . Ainsi,  $f_1$  se comporte comme  $x$ ,  $f_2$  comme  $x^3$  et  $f_4$  comme  $x$ . De plus les constante son petites devant les log et les polynômes petit devant les exponentielles. Donc  $f_3$  se comporte comme  $\log(x)$  et  $f_5$  comme  $2^x$ . En résumé

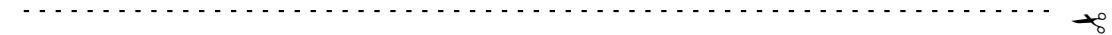
$$f_1 \in \Theta(x), \quad f_2 \in \Theta(x^3), \quad f_3 \in \Theta(\log(x)), \quad f_4 \in \Theta(x), \quad f_5 \in \Theta(2^x)$$

Ainsi

	1	2	3	4	5
1	$\Theta$	$O$	$\times$	$\Theta$	$O$
2	$\times$	$\Theta$	$\times$	$\times$	$O$
3	$O$	$O$	$\Theta$	$O$	$O$
4	$\Theta$	$O$	$\times$	$\Theta$	$O$
5	$\times$	$\times$	$\times$	$\times$	$\Theta$

Remarques :

- Il y a des  $\Theta$  sur la diagonale, car on a toujours  $f \in \Theta(f)$ .
- S'il y a un  $\Theta$  en  $(i, j)$ , il y en a forcément un en  $(j, i)$  car si  $f \in \Theta(g)$  alors  $g \in \Theta(f)$ .
- Il ne peut y avoir à la fois un  $O$  en  $(i, j)$  et en  $(j, i)$ . car si  $f \in O(g)$  et  $g \in O(f)$  alors  $f \in \Theta(g)$  et  $g \in \Theta(f)$ .



### ► Exercice 2. (Calcul de $\lfloor \log_2 \left( \frac{a}{b} \right) \rfloor + 1$ )

On considère l'algorithme suivant :

- En entrée : deux entiers strictement positifs  $a$  et  $b$  tels que  $b \leq 2a$ .
- En sortie :  $N$

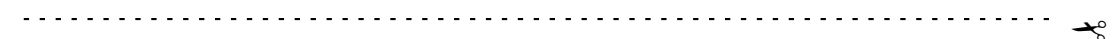
```

1  N <- 0
2  M <- b
3  tant que M <= a faire
4      M <- 2*M
5      N <- N + 1
6  retourner N
```

1. Montrer que l'algorithme termine.



Au départ de la boucle  $M = b$  est un entier strictement positif. À chaque étape de la boucle  $M$  est multiplié par 2. La valeur de  $M$  augmente donc strictement à chaque étape de boucle.  $M$  ne peut donc pas rester indéfiniment inférieur à l'entier  $a$ .



2. Montrer que la condition suivante est vérifiée au début de la boucle et est un invariant de boucle :

$$0 \leq N \quad \text{et} \quad M = 2^N * b \quad \text{et} \quad M \leq 2a.$$



- 
- Au départ de la boucle  $N$  vaut 0 et  $M$  vaut  $b$ . On a donc bien  $0 \leq N$  et  $M = b = 2^0 * b$  et  $M = b < 2a$  par hypothèse.
  - Soit  $m$  et  $n$  les valeurs de  $M$  et  $N$  au début de la ligne 4 et  $m'$  et  $n'$  les valeurs de  $M$  et  $N$  à la fin de la ligne 5. On suppose que l'on exécute une étape de boucle c'est-à-dire que  $m \leq a$  et que l'invariant est vrai au départ, c'est-à-dire que :

$$0 \leq n \quad \text{et} \quad m = 2^n * b \quad \text{et} \quad m \leq 2a.$$

On veut montrer que

$$0 \leq n' \quad \text{et} \quad m' = 2^{n'} * b \quad \text{et} \quad m' \leq 2a.$$

On a

$$m' = 2m \quad \text{et} \quad n' = n + 1.$$

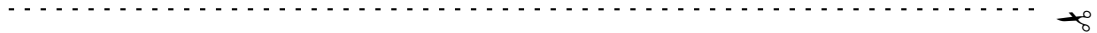
On en déduit que  $0 \leq n \leq n'$ , que

$$m' = 2m = 2(2^n * b) = 2^{n+1}b = 2^{n'} * b,$$

et finalement que

$$m' = 2m \leq 2a.$$

Ainsi, l'invariant est conservé à chaque étape de boucle.



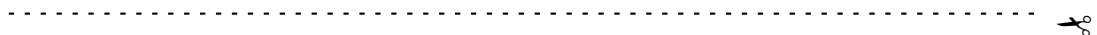
3. En déduire que à la fin de la boucle on a

$$0 \leq N \quad \text{et} \quad M = 2^N * b \quad \text{et} \quad a < M \leq 2a.$$



-----

L'invariant est vrai au départ de la boucle et est conservé à chaque étapes de boucle. Il est donc vrai à la fin. De plus, puisque l'on est sorti de la boucle, c'est que la condition  $M \leq a$  est fausse et donc que  $a < M$ .



4. En déduire que  $N = \lfloor \log_2 \left( \frac{a}{b} \right) \rfloor + 1$ .



-----

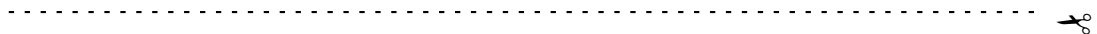
À la sortie de la boucle on a,  $a < 2^N b \leq 2a$ . On a donc d'une part  $\frac{a}{b} < 2^N$  et d'autre part  $2^N \leq \frac{2a}{b}$  c'est à dire  $2^{N-1} \leq \frac{a}{b}$ . Autrement dit :

$$2^{N-1} \leq \frac{a}{b} < 2^N$$

En passant au  $\log_2$  on trouve :

$$N - 1 = \log_2(2^{N-1}) \leq \log_2 \left( \frac{a}{b} \right) < \log_2(2^N) = N.$$

C'est donc que  $N - 1 = \lfloor \log_2 \left( \frac{a}{b} \right) \rfloor$ , soit  $N = \lfloor \log_2 \left( \frac{a}{b} \right) \rfloor + 1$ .

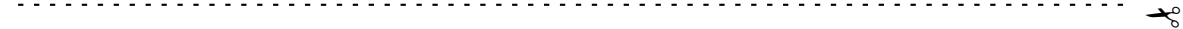


► **Exercice 3. (Nombre de 1 en base 2)**

Écrire un algorithme récursif qui calcule le nombre de chiffres 1 dans l'écriture en base 2 d'un nombre donné  $n$ .



```
.....  
• Algorithme : ChiffreBase2  
• Entrée : un entier positif  $n$   
• Sortie : le nombre de chiffre 1 dans l'écriture en base 2 de  $n$   
  Si  $n = 0$  alors  
    retourner 0  
  sinon  
    retourner  $(n \bmod 2) + \text{ChiffreBase2}(n / 2)$   
• En C (non demandé) :  
  int ChiffreBase2 (int n)  
  {  
    if (n == 0)  
      return 0;  
    else  
      return  $(n \% 2) + \text{ChiffreBase2}(n/2)$ ;  
  }
```



► **Exercice 4. (Fonction inconnue)**

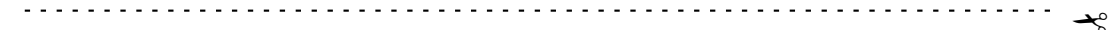
On considère la fonction suivante :

```
1  int syracuse(int n)  
2  {  
3      printf("%i ", n);  
4      if (n == 1) return 1;  
5      else  
6          {  
7              if (n % 2 == 0) return 1+syracuse(n/2);  
8              else          return 2+syracuse(3*n+1);  
9          }  
10 }
```

1. Quel est l'affichage lors de l'appel `syracuse(4)` ? Indication : On montrera que la fonction retourne l'entier 3.



```
.....  
L'affichage est  
4 2 1  
.....
```



2. Même question pour l'appel `syracuse(6)` (la valeur retournée est ici 11) ?



-----  
L'affichage est

6 3 10 5 16 8 4 2 1

Voici la trace de l'exécution :

Appel de `syracuse` avec `n=6`

ligne 3:     `print "%i\n"%n,`

6

ligne 4:     `if (n==1): return 1`

ligne 7:         `if (n%2 == 0): return 1+syracuse(n/2)`

Appel de `syracuse` avec `n=3`

ligne 3:     `print "%i\n"%n,`

3

ligne 4:     `if (n==1): return 1`

ligne 7:         `if (n%2 == 0): return 1+syracuse(n/2)`

ligne 8:         `else:             return 2+syracuse(3*n+1)`

Appel de `syracuse` avec `n=10`

ligne 3:     `print "%i\n"%n,`

10

ligne 4:     `if (n==1): return 1`

ligne 7:         `if (n%2 == 0): return 1+syracuse(n/2)`

Appel de `syracuse` avec `n=5`

ligne 3:     `print "%i\n"%n,`

5

ligne 4:     `if (n==1): return 1`

ligne 7:         `if (n%2 == 0): return 1+syracuse(n/2)`

ligne 8:         `else:             return 2+syracuse(3*n+1)`

Appel de `syracuse` avec `n=16`

ligne 3:     `print "%i\n"%n,`

16

ligne 4:     `if (n==1): return 1`

ligne 7:         `if (n%2 == 0): return 1+syracuse(n/2)`

Appel de `syracuse` avec `n=8`

ligne 3:     `print "%i\n"%n,`

8

ligne 4:     `if (n==1): return 1`

ligne 7:         `if (n%2 == 0): return 1+syracuse(n/2)`

Appel de `syracuse` avec `n=4`

ligne 3:     `print "%i\n"%n,`

4

ligne 4:     `if (n==1): return 1`

ligne 7:         `if (n%2 == 0): return 1+syracuse(n/2)`

Appel de `syracuse` avec `n=2`

ligne 3:     `print "%i\n"%n,`

2

ligne 4:     `if (n==1): return 1`

ligne 7:         `if (n%2 == 0): return 1+syracuse(n/2)`

```
Appel de syracuse avec n=1
ligne 3:      print "%i\n"%n,
1
ligne 4:      if (n==1): return 1
Retour de syracuse avec resultat=1
Retour de syracuse avec resultat=2
Retour de syracuse avec resultat=3
Retour de syracuse avec resultat=4
Retour de syracuse avec resultat=5
Retour de syracuse avec resultat=7
Retour de syracuse avec resultat=8
Retour de syracuse avec resultat=10
Retour de syracuse avec resultat=11
```

----- ✂