Correction interro

Cours d'analyse, algorithmique

—Master 2 CCI—

Durée : 45 min

▸ **Exercice 1. (Notation O et Θ)**

Dans cet exercice f, g sont des fonctions positives.

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

Par définition, $f \in O(g)$ si on peut trouver n_0 et $\alpha > 0$ tel que pour $n \geq n_0$ on a $f(n) \leq \alpha g(n)$. Puisque α est strictement positif, on peut diviser les deux cotés de l'inégalité par α et obtenir $\frac{1}{\alpha} f(n) \leq g(n)$, soit $g(n) \geq \frac{1}{\alpha} f(n)$. En posant, $\beta = \frac{1}{\alpha}$ on a donc pour $n > n_0$, $g(n) \geq \beta f(n)$. C'est la définition de $g \in \Omega(f)$.

2. En déduire que $f \in O(g)$ et $g \in O(f)$ alors $f \in \Theta(g)$

D'après la question précédente, si $g \in O(f)$ alors $f \in \Omega(g)$. Avec $f \in O(g)$, on obtient la définition de $f \in \Theta(g)$.

3. On considère les fonctions suivantes :

$$f_1 = 43x, \quad f_2 = 5x^3 + 8x + 2, \quad f_3 = \log(x), \quad f_4 = 2^x + x, \quad f_5 = 4x + 2.$$

Remplir le tableau suivant en mettant dans la case à l'intersection de la ligne i et de la colonne j le symbole qui convient selon la règle :

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

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

▸ **Exercice 2. (Dessin de rectangle).**

Soit l, h et d trois entiers donnés par l'utilisateur qui désignent respectivement la largeur, la hauteur et la densité d'un rectangle. On demande de dessiner le rectangle correspondant. Pour donner une impression de grisé, on utilise le paramètre de densité de la manière suivante : on affiche le caractère `'*'` tous les d caractères ; on commence par afficher le caractère `'*'` ; les autres caractères sont des `'.'`.

Voici quelques exemples de rectangles ainsi dessinés (les trois nombres au dessus de chaque rectangle sont dans l'ordre l, h et d) :

```
5 5 2         4 5 2       7 5 3             5 4 3
* . * . *     * . * .     * . . * . . *     * . . * .
. * . * .     * . * .     . . * . . * .     . * . . *
* . * . *     * . * .     . * . . * . .     . . * . .
. * . * .     * . * .     * . . * . . *     * . . * .
* . * . *     * . * .     . . * . . * .
```

✂ -

```
nstar = 0
pour i de 1 à h faire
    pour j de 1 a l faire
        si nstar mod d == 0 alors
            afficher("*")
        sinon afficher(".")
        nstar = nstar + 1
    afficher("\n")
```

- ✂

▸ **Exercice 3.** (**Puissances**)

Écrire une *fonction* qui prend en paramètre deux nombres entiers n et p et qui retourne l'exposant de la plus petite puissance de p supérieure ou égale à n. On utilisera une boucle.

Par exemple pour $n = 500$ et $p = 3$ le programme affiche 6 car $3^5 = 243 < 500$ et $3^6 = 729 > 500$.

✂ -

```
ex = 0
puiss = 1
tant que puiss < n faire
    ex = ex + 1
    puiss = puiss * p
retourner ex
```

- ✂

▸ **Exercice 4.** Écrire un programme qui demande un nombre à l'utilisateur et répond si ce nombre est positif, négatif ou nul, ainsi que si ce nombre est divisible par 2 ou 3. Par exemple, si l'utilisateur tape 12 l'ordinateur écrira

```
Le nombre 12 est positif, divisible par 2 et par 3.
```

Si l'on tape 9

```
Le nombre 9 est positif, divisible par 3.
```

Si l'on tape 0

```
Le nombre 0 est nul, divisible par 2 et par 3.
```

Si l'on tape -5

```
Le nombre -5 est négatif.
```

Enfin si l'on tape -8

```
Le nombre -8 est négatif, divisible par 2.
```

On s'efforcera de respecter la sortie précise de ces exemples.

Voici une solution en C avec les affichages précis :

```
#include <stdio.h>
void main(void)
{
  int n;
  scanf("%d", &n);
  printf("Le nombre %d est ", n);
  if (n > 0) printf ("positif");
  else if (n == 0) printf("nul");
  else printf("negatif");
  if (n % 2 == 0)
    if (n % 3 == 0) printf (", divisible par 2 et par 3");
    else            printf (", divisible par 2");
  else
    if (n % 3 == 0) printf (", divisible par 3");
  printf(".\n");
}
```