# TD 1 - Algorithmique

## Exercice 1

Ecrire un algorithme qui permet de lire une année A puis afficher si elle est bissextile ou non (nombre de jours 365 ou 366). On note que : Si A n'est pas divisible par 4, l'année n'est pas bissextile. Si A est divisible par 4, l'année est bissextile sauf si A est divisible par 100 et pas par 400.

## Exercice 2

Ecrire un algorithme qui permet la transformation de coordonnées cartésiennes (x,y) en coordonnées polaires (r,θ) qui se fait par les formules :

$r^2 = x^2+y^2$

si $x > 0$ alors $\theta = \arctan(y/x)$

si $x < 0$ alors $\theta = \arctan(y/x) + \pi$

si $x = 0$ alors $\theta = \pi/2$ si $y > 0$

$\theta = -\pi/2$ si $y < 0$

$\theta = 0$ si $y = 0$

Quel est l'intervalle de θ ? Modifier l'algorithme pour que cet intervalle soit $]-\pi, +\pi]$

## Exercice 3 – Suites récurrentes

La suite de Fibonacci est définie comme suit :

$U_0 = 1$ et $U_1 = 1$

$U_n = U_{n-1} + U_{n-2}$ pour $n \geq 2$

1. Ecrire un algorithme qui permet de lire $n \geq 2$ puis calculer et afficher $U_n$.
2. Ecrire un algorithme qui permet de lire $n \geq 2$ puis calculer et afficher les n premiers termes de la suite.
3. Ecrire un algorithme qui lit un entier A puis affiche le plus petit nombre n pour lequel $U_n > A$.

## Exercice 4

Ecrire un algorithme qui lit une suite de nombres strictement positifs (la fin de la suite est signalée par 0), calcule et affiche la somme des nombres pairs et la somme des nombres impairs de cette suite.

## Exercice 5

Ecrire un algorithme qui réalise la division euclidienne de deux entiers a et b sans utiliser les opérateurs /, * et %. On divise a par b et on afficher le quotient et le reste de la division.

## Exercice 6 – Nombres parfaits

On dit qu'un nombre entier n est parfait s'il est égal à la somme de ses diviseurs stricts positifs. Par exemple 6=1+2+3 est parfait, 28=1+2+4+7+14 est parfait, $8 \neq 1+2+4$ n'est pas parfait.

1. Ecrire un algorithme qui permet de lire un entier positif et de vérifier s'il est parfait ou non.
2. Ecrire un algorithme qui permet d'afficher les n premiers nombres parfaits.

On dit que deux nombres m et n sont amis si la somme des diviseurs stricts de m est égale à n et la somme des diviseurs stricts de n est égale à m.

3. Ecrire un algorithme qui lit deux nombres m et n et teste s'ils sont amis ou non.
4. Ecrire un algorithme qui lit un entier N puis affiche tous les couples amis inférieurs à N.

## Exercice 7 – Nombres premiers

1. Écrire un algorithme qui permet de lire un entier strictement positif n puis vérifier s'il est premier ou non.
2. Ecrire un algorithme qui permet de lire un entier strictement positif n puis afficher les n premiers nombre premiers. Exemple : pour n=4, on affiche les nombre 2, 3, 5 et 7.
3. Ecrire un algorithme qui permet de lire un entier strictement positif n puis afficher tous les nombres premiers inférieurs à n. Exemple : pour n=20, on affiche les nombres 2, 3, 5, 7, 11, 13, 17 et 19.

**Exercice 8**

1. Écrire un algorithme qui permet de lire un entier strictement positif n calculer et afficher le nombre de chiffres de n. Exemple si n=123, le résultat est 3.
2. Écrire un algorithme qui permet de lire un entier strictement positif n calculer et afficher la somme des chiffres de n. Exemple si n=123, le résultat est 6.
3. Ecrire un algorithme qui permet de lire un entier strictement positif n puis calculer et afficher le nombre obtenu en inversant l'ordre des chiffres. Exemple : si l'utilisateur donne 431, le résultat sera : 134.
4. **Racine digitale** : la racine digitale d'un entier n est définie comme le résultat de sommer successivement les chiffres de n jusqu'à trouver un seul chiffre et le résultat est le chiffre obtenu.
   Exemple: la racine digitale de 1729 sera calculée comme suit :
   Etape 1 : $1 + 7 + 2 + 9 = 19$
   Etape 2 : $1 + 9 = 10$
   Etape 3 : $1 + 0 = 1$
   Ainsi la racine digitale de 1729 est 1.
   Ecrire un algorithme qui permet de lire un entier strictement positif n puis calculer et afficher sa racine digitale.

**Exercice 9 – Approximation de la racine carrée**

Soit A un réel positif. Pour calculer $\sqrt{A}$, on utilise la formule récurrente suivante :

$$\begin{cases} X_0 = A \\ X_{n+1} = \frac{1}{2}\left(X_n + \frac{A}{X_n}\right) \text{ pour } n \geq 0 \end{cases}$$

Sachant que $X_n$ converge vers $\sqrt{A}$, écrire un algorithme qui permet de calculer une approximation de $\sqrt{A}$.

**Exercice 10 - Suite de hailstone**

Une suite de hailtone est une suite entière définie en précisant sont terme initial $U_0$ puis on calcule $U_n$ par :

$$U_{n+1} \begin{cases} 1 \text{ si } U_n = 1 \\ \frac{U_n}{2} \text{ si } U_n \text{ est pair} \\ 3U_n + 1 \text{ sinon} \end{cases}$$

Cette suite à toujours tendance de converger vers 1, malgré qu'on n'arrive pas encore à le prouver !

Ecrire un algorithme qui demande la saisie du terme initial de cette suite, puis affiche la séquence de nombre générés par la suite jusqu'à la valeur 1. Exemple : pour $U_0 = 11$ on affiche la séquence :

11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1

*** Exercice 11 - décomposition en facteurs premiers**

Soit n un entier tel que $n \geq 2$. La décomposition de n en facteurs premiers est l'écriture de n sous la forme :

$$n = \prod_{i=1}^{k} {p_i}^{a_i}$$

Où les $p_i$ sont les nombres premiers diviseurs de n.

Ecrire un algorithme qui lit un entier n puis calcule et affiche sur l'écran les $p_i$ et $a_i$ de la décomposition de n.

*** Exercice 12 - Bézout et algorithme d'Euclide étendu**

Soient $a$ et $b$ deux entiers strictement positifs et $p$ le PGCD de $a$ et $b$. Il existe $u$ et $v$ tels que : $u.a + v.b = p$
Etendre l'algorithme d'Euclide pour calculer $p$, $u$ et $v$ pour $a$ et $b$ données.

***Hint*** : Vous pouvez ajouter dans la boucle *while* de l'algorithme d'Euclide deux variables u et v (ou plus selon le besoin) tels que à chaque instant $u.a + v.b = x$. La boucle s'arrête lorsque $x = pgcd$.