Exercices récapitulatifs :

Exercice 1 ★ :

a) Que fait l'algorithme ci-contre ? À l'aide de la méthodologie du cours, réalisez un test afin de montrer que vous avez compris.
b) Déterminez sa complexité.

```
Données :
T : tableau d'entiers de taille 3 ;
i : nombre entier ;
somme : nombre entier
début
    somme=0
    pour i allant de 1 à 3 faire
        somme ← somme + T[i]
    fin
    retourner Valeur de somme
fin
```

Exercice 2 ★ :

a) Écrivez un **algorithme** permettant de savoir si un tableau d'entiers est trié dans l'ordre décroissant.
b) Vous ferez "tourner à la main" votre algorithme en utilisant le tableau T = [17, 9, 8, 11].
c) Vous déterminerez ensuite la complexité de votre algorithme.

Exercice 3 ★ :

a) Écrivez un **algorithme** permettant de trouver le plus grand entier présent dans un tableau.
b) Vous ferez "tourner à la main" votre algorithme en utilisant le tableau T = [3,5,1,8,2].
c) Vous déterminerez ensuite la complexité de votre algorithme.

Exercice 4 ★★ :

On dit que a et b forment un digramme si dans une chaine de caractères, le caractère a est suivi du caractère b (mais pas nécessairement immédiatement).

a) Écrivez un **algorithme** permettant de trouver un digramme dans une chaine de caractères.
b) Vous ferez "tourner à la main" votre algorithme en utilisant la chaine de caractères "Chaton" et en recherchant le digramme formé des lettres "c" et "e".
c) Vous déterminerez ensuite la complexité de votre algorithme.

Exercice 5 ★★★ :

a) Modifiez l'algorithme de l'exercice 3 afin de trouver les deux plus grands entiers présents dans un tableau. Votre algorithme devra être d'une complexité en $\mathcal{O}(n)$.
b) Vous ferez "tourner à la main" votre algorithme en utilisant le tableau T = [3,5,1,8,2,6].

Exercice 6 ★★ :

Dans cet exercice, on s'intéresse à un tableau bidimensionnel T de taille n par n. On accède à la i-ème ligne et à la j-ème colonne de ce tableau grâce à l'instruction :

$$T[i][j]$$

a) Soit le tableau $T = \begin{bmatrix} -5 & 2 & 1 \\ 1 & -4 & 3 \\ 7 & 9 & -5 \end{bmatrix}$.

Quel nombre obtient-on si on demande $T[2][1]$? $T[1][3]$? $T[3][3]$?

b) L'algorithme ci-contre fait la somme de tous les éléments positifs du tableau T. Déroulez cet algorithme sur le tableau T donné ci-dessus.

c) Déterminez la complexité de cet algorithme.

d) Écrire un algorithme calculant la somme des éléments diagonaux du tableau T (on appelle cela la **trace, notée Tr(T)**). Cet algorithme devra avoir une complexité **linéaire**.

```
Données :
T : tableau bidimensionnel d'entiers ;
i, j : nombres entiers ;
somme : nombre entier
début
    somme=0
    pour i allant de 1 à longueur(n) faire
        pour j allant de 1 à longueur(n) faire
            si T[i][j] ≥ 0 alors
                somme ← somme + T[i][j]
            fin
        fin
        retourner Valeur de somme
    fin
fin
```

Exercice 7 ★★ :

Dans cet exercice, on s'intéresse à un tableau bidimensionnel T de taille n par 3. On accède à la i-ème ligne et à la j-ème colonne de ce tableau grâce à l'instruction :

$$T[i][j]$$

a) Compléter l'algorithme ci-contre afin que celui-ci remplace toutes les valeurs négatives du tableau T par des 0.

b) Déterminez la complexité de votre algorithme.

```
Données :
T : tableau bidimensionnel d'entiers ;
i, j : nombres entiers ;
début
    pour i allant de 1 à longueur(n) faire
        pour j allant de 1 à 3 faire
            si ............ alors
                ...............
            fin
        fin
        retourner Valeur de somme
    fin
fin
```

Exercice 8 ★★★ :

Le problème du voyageur de commerce est le suivant :

« étant donné n villes et les distances séparant chaque ville, *trouver un chemin de longueur totale minimale qui passe exactement une fois par chaque ville et revienne au point de départ*. »

On dispose d'une instruction élémentaire « TrouverChemin » qui renvoie la longueur totale d'**un** chemin passant exactement une fois par chaque ville.

Cas particulier : calcul du nombre de chemins pour n = 4

a) Combien de villes peut-on choisir initialement ?
Une fois une ville choisie, combien de villes peut-on choisir ?

b) En déduire le nombre de chemins total dans le cas n = 4 puis le nombre de fois où on appellera l'instruction « TrouverChemin » . On pourra utiliser la notation factorielle définie par $n! = 1 \times 2 \times 3 \times 4 \ldots \times n$.

Cas général : calcul du nombre de chemins pour n grand

a) Combien de villes peut-on choisir initialement ?
Une fois une ville choisie, combien de villes peut-on choisir ?
Répéter l'opération…

b) En déduire le nombre de chemins total dans le cas général puis le nombre de fois où l'on appellera l'instruction « TrouverChemin ». On pourra utiliser la notation factorielle définie par $n! = 1 \times 2 \times 3 \times 4 \ldots \times n$.

c) Conclure sur la complexité de l'algorithme.

d) Dans le cas où l'on met 10 ns (10^{-9} s) pour calculer un chemin et où l'on dispose de 100 villes. Combien de temps va-t-on mettre pour calculer le chemin de longueur minimale?

Extension :

Le choix initial de la ville n'a en fait pas d'importance et nous avons compté les chemins deux fois (aller-retour).

Le nombre d'appels sera donc égal à $\frac{(n-1)!}{2}$. La complexité de l'algorithme va-t-elle changer?