TP2 : Parallélisation de la Multiplication de Matrices avec OpenMP

# Objectifs :

* Comprendre la parallélisation des boucles imbriquées.
* Utiliser OpenMP pour paralléliser la multiplication de matrices.
* Analyser les performances et l'efficacité du code parallèle.

# Matériel requis :

* Un compilateur compatible OpenMP (gcc, clang, etc.).
* Un éditeur de texte ou un IDE.

# Étapes du TP :

## 1. Préparation de l'Environnement

Assurez-vous que votre compilateur supporte OpenMP et que vous avez un éditeur de texte ou un IDE installé (comme décrit dans la réponse précédente).

## 2. Implémentation Séquentielle de la Multiplication de Matrices

1. Écrire un programme C qui effectue la multiplication de deux matrices carrées séquentiellement.
2. Initialiser les matrices avec des valeurs aléatoires ou constantes.

## 3. Parallélisation de la Multiplication de Matrices

1. Utilisez OpenMP pour paralléliser les boucles de multiplication des matrices.
2. Ajouter les directives OpenMP appropriées pour paralléliser les boucles externes.

## 4. Analyse des Performances

1. Comparer les temps d'exécution séquentiel et parallèle.
2. Tester avec différents nombres de threads en utilisant omp\_set\_num\_threads().
3. Documenter les observations et les résultats des tests de performance.
4. Comparer les temps d'exécution pour différents nombres de threads et discuter des résultats.

# Exercices Supplémentaires (optionnels)

## Exercice 1 : Parallélisation Avancée

* Essayez de paralléliser également la boucle interne pour observer les effets sur les performances. Faites attention à l'équilibrage de la charge et aux frais généraux de gestion des threads.

## Exercice 2: Optimisation des Données

* Étudiez l'impact de l'optimisation des accès mémoire, par exemple en changeant l'ordre des boucles pour améliorer la localité des données.

## Exercice 3 : Utilisation de Matrices Non Carrées

* Modifiez le programme pour qu'il fonctionne avec des matrices non carrées et observez l'impact sur les performances.