# INFO632 – TP4

# Compte rendu

Sbai Sami

**Explication du Code :**

**- Structure Voiture :**

Nous avons défini une structure `Voiture` qui stocke les attributs d'une voiture, notamment :

**- id :** Identifiant unique de la voiture.

**- tours :** Nombre de tours déjà effectués par la voiture.

**- Variables Globales Partagées :**

Des variables partagées ont été utilisées pour gérer la course et la synchronisation :

**- voitures\_en\_course :** Compteur des voitures restant en course.

**- classement :** Tableau contenant les voitures classées selon leur ordre d'arrivée.

**- index\_classement :** Index du tableau classement.

**- Routine des Voitures :**

Chaque thread voiture suit cette routine :

**- Départ :** Attendre que toutes les voitures soient prêtes via une barrière (`pthread\_barrier\_wait`).

**- Tours de Piste :** Simuler chaque tour de piste par une attente aléatoire (`usleep`).

**- Notification :** Après chaque tour, verrouiller le mutex, incrémenter le compteur de tours, puis signaler l'arbitre via une variable de condition.

**- Classement :** Lorsque la voiture termine tous ses tours, elle verrouille le mutex, met à jour le classement, et signale l'arbitre.

**- Routine de l'Arbitre :**

La routine de l'arbitre suit ces étapes :

**- Départ :** Attendre que toutes les voitures soient prêtes via une barrière.

**- Suivi de la Course :** En utilisant `pthread\_cond\_wait`, l'arbitre attend les notifications de progression des voitures et les affiche.

**- Classement Final :** Une fois toutes les voitures arrivées, l'arbitre affiche le classement final.

**Résultats Attendus :**

L'utilisateur doit entrer le nombre de voitures qui courent. Chaque voiture annonce sa progression à chaque tour, et lorsqu'elle a terminé la course, elle informe l'arbitre qui affichera le classement final.

