**TP MPI**

**Exercice1 :**

Compilez et exécutez le code 01-txrx.c. Exécutez-le en utilisant 4 processus. Ensuite, modifiez le code pour répondre aux questions suivantes :

1. Exécutez le code en utilisant 5 processus. Que se passe-t-il ? Pouvez-vous l'expliquer ?
2. Modifiez le code pour qu'il fonctionne avec un nombre quelconque de processus.
3. Que se passe-t-il si la valeur du tag dans la fonction MPI\_Recv est différente de MPI\_TAG\_VALUE utilisée par l'expéditeur ? Modifier le code pour que les processus puissent recevoir n'importe quel message indépendamment de la valeur de l'étiquette et de l'expéditeur.
4. Pour les processus qui reçoivent les données, utilisez la variable status pour ﬁndir qui a envoyé le message et imprimez l'id (ou le rang) de l'expéditeur à l'écran.

**Exercice 2 :**

Modifiez le code précédent de sorte que le processus expéditeur envoie un tableau d'entiers alloués dynamiquement (c'est-à-dire en utilisant les fonctions malloc ou calloc). Il y aura autant de récepteurs que d'éléments dans le tableau. Le récepteur de rang 1 recevra le ﬁrst élément du tableau ; le récepteur de rang 2 recevra le deuxième élément, et ainsi de suite.

**Exercice 3 :**

Ecrivez un programme MPI pour effectuer une communication cyclique en anneau sur un nombre pair de processeurs (disons quatre), de sorte que chaque processus envoie son identifiant au processus suivant. Chaque processus affichera à l'écran son propre identifiant ainsi que la valeur reçue. Notez que, puisqu'il s'agit d'une communication cyclique, le dernier processus enverra sa valeur au ﬁrst processus. Notez également qu'avec nos connaissances actuelles de MPI, nous ne pouvons pas réaliser la communication cyclique en une seule étape.

**Exercice 4 :**

En utilisant deux matrices A [NR A] [NC A] et B [NC A] [NCB] allouées dynamiquement, écrire un produit matriciel C = A ∗ B en MPI avec p processeurs. Utilisez une approche maître-esclave selon la procédure suivante :

1. Le processus maître alloue et initialise les matrices.
2. Ensuite, il va diviser les lignes de A en autant de morceaux que de travailleurs (number\_of\_workers = num\_processes - 1). Si le nombre de lignes n'est pas divisible par le nombre de travailleurs, il équilibrera la taille des morceaux autant que possible.
3. Ensuite, le maître enverra à chaque travailleur la taille du chunk (nombre de lignes) ; c'est-à-dire : le chunk correspondant de A et une copie entière de B.
4. Chaque processus de travailleur recevra son morceau de A et une copie de B. Il effectuera le produit matriciel partiel et retournera le résultat au maître.
5. Le maître recevra chaque morceau de C et assemblera le tout.

Vous pouvez utiliser le code 04-mpi\_mm\_skel.c.