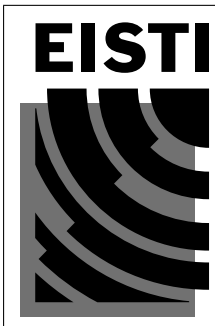	**EX - Examen n°1**
Rachid Chelouah - Juan Angel Lorenzo - Asma Talhi	Architecture et programmation parallèle et répartie
ING2-GSI-MI	Année 2017–2018

Modalités

- Durée : 2 heures.
- Vous devez rédiger votre copie à l'aide d'un stylo à encre exclusivement.
- Toutes vos affaires (sacs, vestes, trousse, etc.) doivent être placées à l'avant de la salle.
- Aucun document n'est autorisé sauf la feuille des fonctions MPI fournie avec ce sujet.
- Tous les codes peuvent être rédigés en C ou C++.
- Aucune question ne peut être posée aux enseignants, posez des hypothèses en cas de doute.
- Aucune machine électronique ne doit se trouver sur vous ou à proximité, même éteinte.
- Aucune sortie n'est autorisée avant une durée incompressible d'une heure.
- Aucun déplacement n'est autorisé.
- Aucun échange, de quelque nature que ce soit, n'est possible.

Questions courtes (4 points)

(1) Expliquez la différence entre les clauses :
'`#pragma omp private(liste de variables)`' et
'`#pragma omp firstprivate(liste de variables)`' (1 pt.) □

(2) Le code OpenMP suivant montre un extrait d'un programme avec un problème de concurrence. Expliquez ce qu'il faut ajouter pour qu'il fonctionne correctement (1 pt.) : □

```
1  # include <omp.h>
2  # include <stdlib.h>
3  # include <iostream>
4  using namespace std;
5
6  int readarray(int *, int);
7  int writearray(int *, int);
8
9  int main(int argc , char *argv []) {
10     int n = 10;
11     int *a = (int *)malloc(n * sizeof(int));
12     int *b = (int *)malloc(n * sizeof(int));
13     omp_set_num_threads(5);
14
15 #pragma omp parallel
16  {
17    int i;
18    int block = n/omp_get_num_threads();
19    int start = omp_get_thread_num()*block;
20    int end = start+block;
21
22    #pragma omp single
23      readarray(b,n);
24
25      for (i=start; i<end; i++)
26          a[i] = b[i] ;
27
28    #pragma omp single
29      writearray(a,n);
```

```
30    }
31  }
```

(3) Expliquez la différence entre les fonctions collectives MPI et leurs versions `"All"`. Par exemple : `MPI_Gather` vs `MPI_Allgather`, `MPI_Reduce` vs `MPI_Allreduce`, etc. (1 pt.) □

(4) Donnez un exemple dans lequel nous pouvons trouver un *deadlock* en MPI et expliquez comment le résoudre (1 pt.) □

Exercice 1 : en OpenMP (4 points)

Écrivez un code OpenMP qui, pour un tableau d'entiers, cherche en parallèle la valeur maximale. À la fin, le master thread devra avoir la valeur maximale.

Exercice 2 : Partage d'un tableau en MPI (4 points)

Nous avons un cluster d'ordinateurs à notre disposition pour calculer le produit des éléments d'un tableau en MPI. Malheureusement, chaque ordinateur a une puissance différente et donc, si nous partageons le tableau en morceaux de même taille, certains ordinateurs mettront plus de temps à finir le calcul. En conséquence, il n'y aura presque aucun avantage d'utiliser la parallélisation. Pour résoudre ce problème, nous avons trié les ordinateurs par ordre de puissance croissante, et nous voulons envoyer à chaque ordinateur un nombre d'éléments du tableau proportionnel à sa puissance. C'est-à-dire, l'ordinateur 1 recevra 2 éléments, l'ordinateur 2 recevra 4, et l'ordinateur i recevra $2 \times i$ éléments du tableau.

- Donnez un code MPI avec p processus qui, **en utilisant des opérations collectives**, calcule le produit des éléments d'un tableau de N éléments en prenant en compte la situation décrite précédemment. Le processus master participera aussi. À la fin du calcul, le processus master recevra les produits partiels et calculera le produit des produits partiels. Nous pouvons considérer que le tableau a la bonne taille pour être réparti entre les processus, c'est-à-dire $N = \sum_{i=1}^{p} 2 * i$.

Exercice 3 : Produit matrix-vecteur (8 points)

Nous voulons calculer un produit matrice-vecteur $A \times X = Y$, où A est une matrice de taille $n \times m$, X un vecteur de taille m et Y le vecteur résultat.

- Donnez le code séquentiel.
- Parallélisez ce code en utilisant l'API OpenMP. **Expliquez clairement** la stratégie suivie pour paralléliser le code.
- Parallélisez le code séquentiel en utilisant MPI. **Expliquez clairement** la stratégie suivie pour paralléliser le code ainsi que le choix des opérations MPI point à point et/ou collectives.