UNIVERSITÉ D'ORLÉANS **Programmation Parallèle**
M1 Info Année **2019 - 2020**

Série de Travaux Dirigés : **2** *- MPI - Distribution et Réduction*

Exercice 1. Calcul du max et de sa position dans un tableau

A partir du squelette fourni dans *TD2.tgz* complétez le code pour

- Distribuer le tableau généré sur le processeur `root` à l'ensemble des processeurs et sans faire d'hypothèse sur la taille du tableau.
- Calculer le max en local en utilisant la structure `max_loc` afin de fournir également la position du max.
- Finir le programme principal avec un `MPI_Reduce` afin d'obtenir sur le processeur `root` le résultat final.

Le programme principal proposé dans `max.cpp` prend 2 arguments la taille du tableau global et le processeur `root`.

Exercice 2. Suite de Syracuse

Une suite de Syracuse est une suite telle que $U_0 = x$ avec $x > 0$ et

$$U_i = \begin{cases} \frac{U_{i-1}}{2} & si\ U_{i-1}\ est\ pair \\ 3U_{i-1}+1 & sinon \end{cases}$$

On souhaite vérifier si un tableau d'entiers de taille n correspond à une suite de Syracuse alors que ce tableau est initialement sur le processeur `root`.

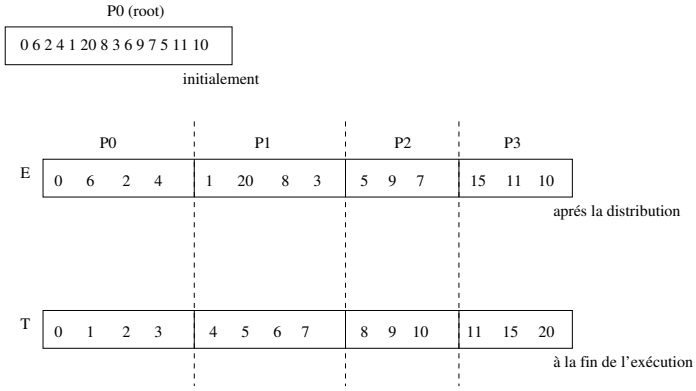
1. Décrivez la parallélisation que vous envisagez sachant qu'on souhaite le cas général lorsque n n'est pas divisible par `root`.
2. Explicitez les fonctions de communications MPI que vous allez utiliser et pourquoi.
3. Proposez une implémentation à partir du squelette disponible sous Celene sachant que le programme prend 4 arguments en ligne de commande
   - la taille du tableau global,
   - le processeur `root`,
   - la graine pour la génération aléatoire
   - le type de génération (0 : complétement aléatoire, 1 : partiellement syracuse et 2 : syracuse).

Exercice 3. Parallélisation du tri par induction

Soit l'algorithme suivant qui réalise un tri appelé tri par induction

```
1 for i from 1 to n do
2   p = 1
3   for j from 1 to n do
4     if (E(j)<E(i)) then
5       p = p + 1
6     end if
7   end for
8   T(p) = E(i)
9 end for
```

Ainsi si E est un tableau d'entiers tous distincts, on souhaite paralléliser la première boucle for (indice i) en répartissant le calcul des positions p sur les processeurs. Initialement le tableau E est généré par un processeur `root` et distribué aux autres. A la fin du tri, on souhaite que le tableau résultat reste réparti sur les `nprocs` processeurs.



1. Définissez les étapes de la parallélisation sachant que le tableau E est initialement généré par le processeur `root`.
2. Par rapport à ces étapes, explicitez les fonctions de communications MPI que vous allez utiliser.
3. A partir du squelette disponible sous Celene complétez l'implémentation avec toujours 2 arguments pour le programme principal, la taille du tableau global et le processeur `root`.