Exercices INF8541: série #14

1. Mécanismes de synchronisation et communication

On veut définir une machine, qui peut être utilisée de façon *concurrente* par plusieurs clients, pour gérer une liste de nombres et trouver le maximum. Initialement, la liste associée à la machine ne contient aucun nombre. Les deux opérations exportées par le module sont les suivantes:

- `ajouter( int val )`: Ajoute la valeur `val` dans la liste.
- `trouver_max( int* max )`: Retourne la valeur maximum parmi les diverses valeurs présentes dans la liste. Si aucune valeur n'est présente dans la liste, l'opération est suspendue (bloquée) jusqu'à ce qu'une valeur soit ajoutée. Après l'appel à `trouver_max`, la liste est à nouveau vide.

Vous devez définir différentes versions de ce module:

1. Un module utilisable par l'intermédiaire de RPC et utilisant des sémaphores pour assurer l'exclusion mutuelle et les synchronisations conditionnelles (section 8.1)
2. Un module utilisant l'approche par rendez-vous (section 8.2).

Dans chaque cas, vous devez évidemment vous assurer que le comportement est correct en présence de requêtes multiples possiblement faites de façon concurrente.

Note importante: Si plusieurs requêtes pour obtenir le maximum sont en attente et qu'un ajout est effectué, vous pouvez utiliser *n'importe quelle stratégie* pour sélectionner la requête qui sera satisfaite; vous n'avez donc pas à respecter l'ordre d'arrivée des requêtes pour `trouver_max`.

Note: Vous pouvez utiliser les types de la bibliothèque de Hanson ("*C Interfaces and Implementations*") pour réaliser vos opérations.