Exercices INF8541: série #6

1. Barrière (basé sur l'exercice 3.16 de Andrews)

Soit le code suivant qui réalise une barrière pour `n` processus:

```
int arrive[1:n] = ([n] 0);   # Tableau partage par les n processus

process Worker[1] {
   while(true) {
     # ... code pour la tache 1 ...
     arrive[1] = 1;
     < await (arrive[n] == 1); >
   }
}

process Worker[i = 2 to n] {
   while(true) {
     # ... code pour la tache i ...
     < await (arrive[i-1] == 1); >
     arrive[i] = 1;
     < await (arrive[n] == 1); >
   }
}
```

1. Expliquer de quelle façon fonctionne cette barrière, si on ignore la présence de la boucle `while`.
2. De quelle façon devrait-on modifier le code pour la rendre réutilisable (i.e., pour tenir compte de la boucle)?

2. Sac de tâches (notation Andrews)

On veur définir une procédure `nb_occurences` qui va permetre de déterminer le nombre d'occurences d'un élément `val` dans un tableau `elems` (de taille `n`). L'interface de cette procédure est la suivante:

```
int nb_occurences( int elems[], int n, int val );
/* Effet:
 *    resultat = nombre d'occurences de val dans le tableau elems;
 */
```

En utilisant la notation d'Andrews, écrivez une version de cette procédure qui utilisera une approche de style "sac de tâches" pour générer le parallélisme et produire le résultat, où chaque sous-tâche va travailler sur un intervalle de taille `K`. Vous pouvez supposer que `K` est un identificateur global accessible à la procédure `nb_occurences`.

Vous pouvez évidemment introduire, si nécessaire, une ou plusieurs procédures auxiliaires. Finalement, vous pouvez aussi supposer que `n` est divisible par `K`.