Exercices INF8541: série #9

1. Produit scalaire de deux vecteurs

Le processus `Maitre`, présenté plus bas, définit 2 tableaux `a` et `b`, ainsi qu'une variable `n` indiquant la taille de ces tableaux. En utilisant les canaux introduits un peu plus bas, complétez le code du processus `Maitre` (mais en ignorant le code pour définir `a` et `b`) ainsi que le code du processus `Calculer_ps`. Remarquez qu'un nombre fixe (`N`) de processus `Calculer_ps` est introduit et que chacun de ces processus doit recevoir une série de valeurs qui correspond à un sous-intervalle du tableau de taille `n/N`.

```
chan taille[N](int),      # Canaux pour transmettre la taille du tableau.
     elems[N](int, int),  # Canaux pour transmettre les elements des tableaux.
     resultat(int);       # Canaux pour transmettre un resultat intermediaire.

# Borne inferieure de l'intervalle pour le processus Calculer_ps[i]
int inf( int i, int n ) { return( i * n / N ); }

process Calculer_ps[i = 0 to N-1]
{









}

process Maitre
{
  int i, v, ps, n;
  int a[], b[];
  # Code (omis) pour lire/definir n, a, et b.
  # ...

  # Transmission des donnees aux divers processus: on envoie, dans chaque
  #  message, un (1!) element de chacun des tableaux.
  # A completer.









  # Reception des resultats et cumul avant impression.
  ps = 0;
  for [i = 0 to N-1] {
    receive resultat(v);
    ps += v;
  }
  printf( "Le produit scalaire de a et b est %d\n", ps );
}
```

2. Factoriel

Complétez le code, dans la notation d'Andrews, d'un processus `Fact` pour lequel on aura `N` instances (avec `N <= n`) et qui permettront de calculer `n!`, la valeur `n` étant une valeur lue par le processus 0 à l'aide de l'opération `scanf("%d", &n)`, valeur ensuite transmise à chacun des autres processus.

Le travail consistant à multiplier les valeurs doit être décomposé entre les `N` processus de façon relativement uniforme (décomposition statique, où vous pouvez supposer que `n % N == 0`). C'est le processus 0 qui doit ensuite imprimer le résultat final, après avoir combiné les réponses reçues par les autres instances de processus. Le programme, qui sera donc de type SPMD, devrait utiliser les (tableaux de) canaux indiqués plus bas (voir figure en classe).

```
chan valeur_n[N](int),    # Pour transmettre la valeur de n.
     resultat(int);       # Pour transmettre les resultats des sous-problemes.

process Fact[i = 0 to N-1]
{









}
```