Exercices INF8541: série #4

Le programme Threaded-C présenté plus bas construit un arbre binaire dont les noeuds sont *distribués* sur les différents processeurs, puis parcourt cet arbre pour calculer la somme des éléments contenus dans les noeuds et feuilles de l'arbre. Écrivez le code pour la procédure `calculer_somme`.

```
#include "PRELUDE.h"

/* Les types pour definir les arbres binaires. */
typedef enum { Feuille, Noeud } Sorte;
typedef struct Arbre *          Arbre;
typedef struct Arbre *GLOBAL    Arbre_G;

struct Arbre {
  Sorte sorte;
  union {
    /* sorte == Feuille. */
    struct {
      int val;
    } feuille;

    /* sorte == Noeud. */
    struct {
      int val;
      Arbre_G gauche;
      Arbre_G droite;
    } noeud;
  } info;
};


/* Procedure que vous devez definir. */
THREADED calculer_somme( Arbre_G a, int *GLOBAL res, SPTR done );


THREADED construire_arbre( int n, Arbre_G *GLOBAL res, SPTR done )
{
  /* Construit un arbre de hauteur n distribue' sur les differents
     processeurs.

     Si n <= 1, alors une simple feuille est produite.
     Si n >  1, alors un noeud avec deux enfants est produit.

     Pour etre certain que le bon resultat est produit lorsqu'on fera la
     somme, le champ val est toujours defini comme etant 1.

     Retourne le pointeur resultant dans res et signale done. */
```

```
  Arbre a;

  MALLOC( a, struct Arbre );
  if (n <= 1) {
    a->sorte = Feuille;
    a->info.feuille.val = 1;
    SPAWN( RETOURNER_RES );
  } else {
    a->sorte = Noeud;
    a->info.noeud.val = 1;
    TOKEN( construire_arbre, n-1,
           TO_GLOBAL(&a->info.noeud.gauche), TO_SPTR(RETOURNER_RES) );
    TOKEN( construire_arbre, n-1,
           TO_GLOBAL(&a->info.noeud.droite), TO_SPTR(RETOURNER_RES) );

  }

  FIBER RETOURNER_RES <* 2 *> {
    PUT_SYNC( TO_GLOBAL(a), res, done );
    TERMINATE;
  }
}


void imprimer_usage_et_terminer( char *nom_prog )
{
  printf( "usage:\n  %s n (avec n >= 1)\n", nom_prog );
  printf( "\n" );
  printf( " OU n indique le nombre de niveaux requis pour l'arbre\n" );
  exit(-1);
}


THREADED MAIN( int argc, char* argv[] )
{

  int n;
  Arbre_G a;
  int somme;

  /* On lit l'argument a utiliser. */
  if ( argc <= 1 )
    imprimer_usage_et_terminer(argv[0]);
  n = atoi(argv[1]);
  if ( n <= 0 )
    imprimer_usage_et_terminer(argv[0]);

  /* On construit l'arbre binaire. */
  TOKEN( construire_arbre, n, TO_GLOBAL(&a), TO_SPTR(CALCULER_SOMME) );

  /* On parcourt l'arbre pour calculer la somme des valeurs qu'il contient. */
  FIBER CALCULER_SOMME <* 1 *> {
    INVOKE( OWNER_OF(a), calculer_somme, a, TO_GLOBAL(&somme), TO_SPTR(IMPRIMER) );
  }

  /* On imprime la somme resultante. */
  FIBER IMPRIMER <* 1 *> {
    printf( "somme = %d\n", somme );
    TERMINATE;
  }
}
```