Exercices INF8541: série #13

1. I-structures avec rendez-vous

Comme on l'a déjà vu dans des exercices précédents, une I-structure est une structure de données utilisée pour la programmation parallèle et permettant la synchronisation implicite entre producteur et consommateur(s). Plus précisément (rappel), une I-structure supporte les opérations suivantes:

- `void I_ALLOC( int taille )`: Alloue la I-structure avec `taille` éléments, tous vides.
- `void I_WRITE( int index, T val, bool *ok )`: Écrit la valeur `val` à la position `index`. Si cette position a déjà été remplie, une erreur est signalée (`*ok == FALSE`). Si des requêtes de lecture sont en attente, alors elles sont réactivées.
- `void I_READ( int index, T *val_lue )`: Retourne le contenu de la I-structure à l'`index` indiqué dans `*val_lue`. Si la cellule associée est vide, alors la requête est suspendue jusqu'à ce qu'une écriture appropriée soit effectuée.

Complétez, dans la notation d'Andrews et en utilisant une approche *par rendez-vous*, la mise en oeuvre d'un module encapsulant une I-structure, tel que présenté plus bas.

Assurez-vous que les opérations de lecture et écriture ne puissent pas être utilisées tant que la I-structure n'a pas été initialisée avec l'opération `I_ALLOC`. Assurez-vous aussi qu'il ne soit pas possible, une fois la I-structure allouée, d'appeler de nouveau l'opération `I_ALLOC`.

```
typedef struct {
  bool vide;          # Indique si la cellule est vide ou non.
  T contenu;          # Valeur conservee par la cellule (si pas vide).
} I_struct_item;

typedef I_struct_item *I_STRUCTURE;


module I_structure
  op I_ALLOC( int taille ),
  op I_READ ( int index, T *val_lue ),
  op I_WRITE( int index, T val, bool *ok )
body
  process Gestionnaire_I_structure {
    # A completer.









  }
end I_structure
```