Test de système d'exploitation 2

Nom: ..... Prénom:..... Groupe: .....

Nom: ..... Prénom:..... Groupe: .....

Nous souhaitons implémenter en C le modèle producteur/consommateur où plusieurs threads producteurs et plusieurs threads consommateurs partagent un même buffer et chaque thread producteur/consommateur produit plusieurs items. Ci-dessous le programme que nous vous demandons de compléter.

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>

#define BUFFER_SIZE 5          /* Taille buffer */
#define NP          3          /* Nombre de producteurs */
#define NC          3          /* Nombre de consommateurs */
#define NITERS      4          /* Nombre d'items produits/consommes */

typedef struct {
    int buffer[BUFFER_SIZE];   /* Buffer partage */
    int in;                 /* buffer[in%BUFFER_SIZE] est la premiere case vide */
    int out;               /*buffer[out%BUFFER_SIZE] est la premiere case pleine */
    sem_t full;            /* Represente le nombre de cases pleines */
    sem_t empty;           /* Représente le nombre de cases vides */
    sem_t mutex;           /* Pour l'exclusion mutuelle a la variable partagee */
} sbuf_t;

sbuf_t shared;

void *producteur(void *arg) {
    int i, item, index;
    index = (int)arg;
    for (i=0; i < NITERS; i++) {
        /* Produire un item */
        item = i;

        /* Preparer le depot de l'item dans buffer */
        /* S'il n'y a pas de cases vides, attendre */
        sem_wait(&shared.empty);

         /* Si un autre thread (producteur ou consommateur) utilise buffer,
attendre */
        ................................

        /* Depot de l'item */
        shared.buffer[shared.in] = item;

        shared.in = (shared.in+1)%BUFFER_SIZE;
        printf("[P%d] Produit %d ...\n", index, item); fflush(stdout);

        /* Liberer le buffer */
        ................................

        /* Incrementer le nombre de cases pleines */
        sem_post(&shared.full);

        /* Entrelacer l'execution du producteur et du consommateur */
```

```
        if (i % 2 == 1) sleep(rand()%4);
    }
}

void *consommateur(void *arg) {
    /* Inserer ici le code du consommateur */









}

int main()
{
    srand(time(NULL));
    pthread_t idP, idC;
    int index;

    /* Initialisation des semaphores full et empty */
    sem_init(&shared.full, ...., ...);
    sem_init(&shared.empty, ..., ...);

    /* Initialisation de mutex*/
    ....................................

    for (index = 0; index < NP; index++)
    {
       /* Creer un nouveau producteur */
       pthread_create(&idP, NULL, producteur, (void*)index);
    }

    /* Inserer ici le code de creation de NC consommateurs */
    ....................................




    /* Detruire les semaphores full et empty */
    ................................

    /* Detruire mutex */
    ................................

    /* Terminer le thread main */
    pthread_exit(0);
}
```