Rappels des fonctions de manipulation des threads

1. #include <pthread.h>
2. int **pthread_create** (pthread_t *thread, const pthread_attr_t *attr, void *(*routine)(void*), void *arg);: La fonction **pthread_create** crée un thread, et renvoie 0 si la création s'est bien déroulée, ou le code de l'erreur sinon. Elle reçoit en argument suivant: (1) le TID (Thread Identifier) du thread, (2) une constante,(3) un pointeur vers la fonction exécutée par le thread et (4) un argument de la fonction.
3. int **pthread_join** (pthread_t *thread, void **value_ptr) ; La fonction **pthread_join** bloque l'appelant en attente de la fin du thread passé en 1er argument, alors que le 2nd argument est un pointeur de pointeur qui servira à récupérer l'adresse de la valeur renvoyée par le thread_exit du thread, cette fonction renvoie 0 si le thread se termine correctement, sinon elle renvoie le code de l'erreur

---

Exemple

```
#include < stdio.h >
#include < stdlib.h >
#include < unistd.h >
#include < pthread.h >
void* A(void* arg){
      printf("Tache 1: \n");
}
void* B(void* arg){
      printf("Tache 2: \n");
}
int main(void) {
    pthread_t tache1, tache2;
    pthread_create (&tache1, NULL, &A, NULL);
    pthread_create (&tache2, NULL, &B, NULL);
    pthread_join(tache1, NULL);
    pthread_join(tache2, NULL);
    return 0; }
```

Exercice 1 (Problème des accès concurrents) : Écrire un programme qui initialise une variable entière *value_globale* à 1 et crée deux thread *th_a* et *th_b*. Le thread *th_a* incrémente *value_globale* 100000 fois et le thread *th_b* décrémente *value_globale* 100000 fois.

1. Quelle devrait être la valeur finale de *value_globale* ?

2. Que remarquez-vous après plusieurs exécutions du programme ?

3. Expliquez