Exercice1 :

Implémentez l'algorithme de Peterson pour l'exclusion mutuelle par attente active dans le cadre de 2 threads.
Elaborez ensuite un scénario afin de tester que votre implémentation fonctionne correctement.

- Que remarquez-vous ?
- Si le résultat que vous obtenez n'est pas celui escompté, essayez de trouver un moyen pour corriger le problème, peut être s'orienter vers l'utilisation `__sync_synchronize();`

Exercice2

Le compilateur gcc offre la fonction suivante qui est garantie d'être atomique :

```
type __sync_val_compare_and_swap(
     type *ptr, type oldval, type newval)
```

Cette fonction implémente, de façon **atomique**, l'équivalent des instructions ci-dessous :

```
int compare_and_swap(int *reg, int oldval, int newval) {
        int old_reg_val = *reg;
        if (old_reg_val == oldval)
                *reg = newval;
        return old_reg_val;
}
```

Nous aimerions réaliser un mécanisme d'exclusion par attente active à l'aide de la fonction `__sync_val_compare_and_swap`.

Voici le squelette de ce que nous aimerions implémenter :

```
struct  lock_st {
     // Structure à remplir
};
typedef struct lock_st lock_t;

// Cette fonction initialise le verrou passé en argument
void init_lock(lock_t *lock) {
     // A remplir
}

// Cette fonction verrouille (acquiert) le verrou passé en argument
void acquire_lock(lock_t *lock) {
     // A remplir
}

// Cette fonction déverrouille (relâche) le verrou passé en argument
void release_lock(lock_t *lock) {
     // A remplir
}
```

Pour cet exercice, il est demandé :

1. D'implémentez le code ci-dessus (structure et fonctions)
2. De vérifier que l'implémentation fonctionne correctement verrouillant un compteur incrémenté par 2 threads de manière concurrente.