修改部份

```c
void p0(void) {
    printf("start p0\n");
    while (1) {
        //Peteron's solution的進去部分的程式碼
        //atomic_store(&flag[0], 1);
        atomic_store_explicit(&flag[0],1,memory_order_relaxed); //B
        atomic_thread_fence(memory_order_seq_cst);
        atomic_store_explicit(&turn,1,memory_order_relaxed);
        //atomic_store(&turn, 1);
        while (atomic_load(&flag[1]) && atomic_load(&turn)==1)
                ;   //waiting
        //底下程式碼用於模擬在critical section
        cpu_p0 = sched_getcpu();
        in_cs++;    //計算有多少人在CS中
        nanosleep(&ts, NULL);
        if (in_cs == 2) fprintf(stderr, "p0及p1都在critical section\n");
        p0_in_cs++; //P0在CS幾次
        nanosleep(&ts, NULL);
        in_cs--;    //計算有多少人在CS中
        //Peteron's solution的離開部分的程式碼
        atomic_store(&flag[0], 0);
    }
}
```

```c
void p1(void) {
    printf("start p1\n");
    while (1) {
        //atomic_store(&flag[1], 1);
        atomic_store_explicit(&flag[1],1,memory_order_relaxed);
        atomic_thread_fence(memory_order_seq_cst);
        //atomic_store(&turn, 0);
        atomic_store_explicit(&turn,0,memory_order_relaxed);
        while (atomic_load(&flag[0]) && atomic_load(&turn)==0)
                ;   //waiting
        //底下程式碼用於模擬在critical section
        cpu_p1 = sched_getcpu();
        in_cs++;
        nanosleep(&ts, NULL);
        if (in_cs == 2) fprintf(stderr, "p0及p1都在critical section\n");
        p1_in_cs++;
        nanosleep(&ts, NULL);
        in_cs--;
        //離開critical section
        atomic_store(&flag[1], 0);
    }
}
```

證明：

1. mutual exclusive

使用了memory_order_relaxed來讓P0 P1共享turn，確保兩者看到的值一致，不是0就是1，不用的話可能P0看到1，P1看到0

當 P0跟P1同時想進入時，由於turn被限制住了，所以只會有一個task能進去

2. bounded waiting

P0從cs出來後要再進去一次時，會設定flag[0] = 1，turn = 1，並會禮讓P1先進去cs

3. progress

三種情況：

1. P0想進 P1不想進
   flag[0] = 1, flag[1] = 0, turn = 1.
   while(flag[1] == true && turn == 1); 不成立
   當下沒有task在cs裡，P0進入cs
2. P0 P1 都想進，然而P0僅執行到flag[1] = 1
   當下情況 -> flag[0] = 1, flag[1] = 1, turn = 1
   由於turn = 1
   則P0的while loop成立，並等待P1離開cs且讓flag[1]=0
3. P0 P1都想進，並且flag跟turn皆設定完畢
   則由於memory_order_relaxed共享turn，turn有唯一性，只有一個task能進入cs