Work Sheet 5.1: HW Solutions to the Critical-Section Problem

Consider the following solution to the **Critical Section Problem** using the **test_and_set** hardware instruction.

```
for (iter = 1; iter<=2; iter++) {
1    waiting[i] = true;
2    key = true;
     printf("\n Process %d is waiting", i);
3    while (waiting[i] && key)
4        key = test_and_set(&lock);
5    waiting[i] = false;

printf("\n Process %d enters CS", i);

     /* critical section */

6   j = (i + 1) % n;
7   while ((j != i) && !waiting[j])
8        j = (j + 1) % n;
printf("\n j = %d", j);
9    if (j == i)
10     lock = false;
11  else
12     waiting[j] = false;

   /* remainder section */

}
```

1. Assume that there are six processes (P_0, P_1, …, P_5). Considering the print statements added to the above code, show the output printed by the code if the following sequence of events takes place (the order is very important):
   P_5 executes Line 3 in its first iteration
   P_4 executes Line 3 in its first iteration
   P_0 executes Line 3 in its first iteration

   Assume that the remainder section is so short that if P_x exits its critical section and P_y enters its critical section next, P_x will complete its remainder section and execute Line 3 in its second iteration before P_y exits its critical section. So, when P_y exits its critical section, P_x will be waiting to enter the critical section for the second time.

   Note: Parenthesized text is for explanation only (it does not actually appear in the output)

   Process 5 is waiting

   Process 5 enters CS
   Process 4 is waiting
   Process 0 is waiting
   j = 0 (because the loop on Lines 7 and 8 searches in the order 0, 1, 2, 3, 4, 5)

   Process 0 enters CS
   Process 5 is waiting (for its second turn)
   j = 4 (because the loop on Lines 7 and 8 searches in the order 1, 2, 3, 4, 5, 0)

Process 4 enters CS
Process 0 is waiting (for its second turn)
j = 5 (because the loop on Lines 7 and 8 searches in the order 5, 0, 1, 2, 3, 4)

Process 5 enters CS (for the second time)
Process 4 is waiting (for its second turn)
j = 0

Process 0 enters CS (for the second time)
j = 4

Process 4 enters CS (for the second time)
j = 4 (no processes are waiting)

2. Can the above solution cause starvation? If yes, give a scenario (sequence of events) that causes starvation. If not, explain why. Of course, you must answer this question for the general case where each process may request access to the critical section an arbitrary number of times, not only two times. (**Limit: 3 lines**).

   Of course, it won't, because when a process is done with the CS, it checks to see if there are waiting processes and will give the CS to one of them. So, a process won't take the CS for a second time unless no other process is waiting.

3. If the total number of processes is **n**, what's the maximum number of other processes that a waiting process may wait for before entering the critical section?
   n-1