7.5

a. 增加可用资源在任何情况下都安全
b. 减少可用资源可能会导致死锁（系统的安全状态假设有一定数量的可用资源）
c. 可能会导致死锁。增加一个进程的最大需求意味着系统必须准备更多资源来满足该进程的潜在需求，可能导致系统进入不安全状态。
d. 安全，不会导致死锁
e. 可能会导致死锁。为了保持系统的安全性，必须确保新进程的最大资源需求与当前资源分配不会将系统推向不安全状态（死锁）。
f. 安全，不会导致死锁

7.6

假设系统发生了死锁。这意味着每个进程都占有一个资源，并且还在等待另一个资源。
由于有三个进程和四个资源，所以至少有一个进程能够获得两个资源。这个进程不再需要其他资源，因此在完成后会归还其资源。
因此，系统不会进入死锁状态。

7.11

a. Need:

	A	B	C	D
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

b.
当前的 *Available* 为 1,5,2,0，可以满足 P_3，P_3 完成后释放资源， *Available* 为 1,11,5,2，可以满足剩下的进程。故处于安全状态。

c.
满足该需求后 *Available* 为 1,1,0,0，可以按照顺序：P_0, P_2, P_3, P_4, P_1 执行
故该需求可以被满足。

7.14

```c
semaphore ok_to_cross = 1;

void enter_bridge() {
    ok_to_cross.wait();    // 尝试进入桥时，等待信号量变为1
}

void exit_bridge() {
    ok_to_cross.signal(); // 离开桥时，释放信号量
}
```

- **信号量** `ok_to_cross`：它控制进入桥的许可，初始值为 1（表示桥上没有任何方向的农夫，可以允许一个方向的农夫进入）。
- **进入桥**：当农夫调用 `enter_bridge()` 时，执行 `ok_to_cross.wait()`。如果信号量值为 1，那么农夫获得进入桥的许可，信号量变为 0；如果信号量值已经是 0，意味着桥上已经有农夫，那么其他农夫必须等待，直到桥上的农夫离开桥并释放信号量。
- **离开桥**：当农夫调用 `exit_bridge()` 时，执行 `ok_to_cross.signal()`，将信号量值恢复为 1，表示桥现在空了，允许其他方向的农夫进入。