Solutions
15-410, Fall 2016, Homework Assignment 1.

1 Chefs (5 pts.)

...The maximum numbers they need are shown in the following table.

Chef	Knives	Bowls	Woks
Jamie	2	2	3
Kelly	3	2	3
Morgan	2	2	3

Imagine that the kitchen contains 5 knives, 6 bowls, and 6 woks.
Consider the following system state.

Chef	Knives	Bowls	Woks
Jamie	1	2	2
Kelly	1	0	1
Morgan	1	1	2
Available	2	3	1

1.1 2 pts

Is the state shown above safe, unsafe, or deadlocked? Explain.

First, it is not possible to say that the system is deadlocked because a deadlock requires four ingredients, one of which is a cycle in a wait graph, and requests are not part of a system state as depicted above.

This system is in a safe state because:

1. Jamie can be satisfied if given one knife and one wok, and the system has two free knives and one free wok. When Jamie is done the system will have three free knives, five free bowls, and three free woks.
2. If the system has available (3, 5, 3) then it is easy to satisfy Kelly, who already has (1, 0, 1) and thus needs only (2, 2, 2) to run to completion, leaving free resources of (4, 5, 4).
3. If the system has available (4, 5, 4) it is easy to satisfy Morgan, who already has (1, 1, 2) and thus needs only (1, 1, 1) to run to completion.

In other words, the system is in a safe state because the sequence "Jamie; Kelly; Morgan" is safe.

1.2 3 pts

It is the case for at least one chef that, for some tool type, a request for one more of that tool, if granted, would leave the system in a safe state, but a request for two more of that tool, if granted, would leave the system in an unsafe state. State the two requests that the chef is entitled to make and justify your claim that one of the requests, if granted, would leave the system in a safe state but the other request, if granted, would leave the system in an unsafe state.

If Kelly asks for one more knife, that request, if granted, would put the system in this state:

Chef	Knives	Bowls	Woks
Jamie	1	2	2
Kelly	2	0	1
Morgan	1	1	2
Available	1	3	1

This system is in a safe state because: Jamie can be satisfied if given one knife and one wok, and the system has one free knife and one free wok. When Jamie is done the system will have three free knives, five free bowls, and three free woks. This is the same state as was shown above to be safe (the safe sequence for that state is "Kelly; Morgan."

If instead Kelly asks for *two* knives, which is legal, the system would be in this state:

Chef	Knives	Bowls	Woks
Jamie	1	2	2
Kelly	3	0	1
Morgan	1	1	2
Available	0	3	1

At this point the system has 0 knives. Kelly is entitled to ask for 2 more bowls, and the system has 3, so no problem there. Kelly is also entitled to ask for 2 more woks, but the system has only 1, so Kelly can't necessarily run to completion, so no safe sequence can begin with Kelly.

Meanwhile, both Jamie and Morgan are entitled to ask for more knives, but there are no more knives, so neither of them can necessarily run to completion, so no safe sequence can begin with Jamie or Morgan.

If no safe sequence can have a first step, no safe sequence exists.

2 "Carpe Diem" (5 pts.)

```
       volatile int want[2] = {0, 0};
       volatile int turn = 0;

 1.    do {
 2.       ...remainder section...
 3.       want[i] = 1;
 4.       while (turn != i) {
 5.         while (want[j])
 6.           continue;
 7.         turn = i; // carpe diem
 8.       }
 9.       ...begin critical section...
10.       ...end critical section...
11.       want[i] = 0;
12.    } while (1);
```

There is a problem with this critical-section protocol. Identify a required property which this protocol does not have and then present a trace which supports your claim.

This critical-section protocol doesn't ensure mutual exclusion.

Execution Trace

time	Thread 0	Thread 1
0		3: want[1]=1
1		4: while (turn != 1) // true
2		5: while (want[0]) // false
3	3: want[0]=1	
4	4: while (turn != 0) // false	
5	9: ...enter...	
6		7: turn=1 // carpe diem
7		9: ...enter...

By the way, the code above is known as "Hyman's Solution"—it was mistakenly published in CACM in 1966.

If you believe this protocol also doesn't ensure one of the other properties, you might be right. Just make sure that all of the things you write in your trace can actually happen. In particular, if you assert a loop will run indefinitely, it must be the case that the loop condition obviously can be true indefinitely. *In other words, do not show something happening once and write "this can happen indefinitely" if it can't happen indefinitely because the first time through some state is changed.*

In terms of difficulty, this problem is probably a little easier than an exam question on a similar topic. However, this question represents the sort of reasoning we expect you to carry out.