15-410, Fall 2017, Homework Assignment 1.

Due Wednesday, October 11, 20:59:59 p.m.

Please observe the non-standard submission time... As we intend to make solutions available on the web site immediately thereafter for exam-study purposes, please turn your solutions in on time.

Homework must be submitted in either PostScript or PDF format (not: Microsoft Word, Word Perfect, Apple Works, LaTeX, XyWrite, WordStar, etc.). Submit your answers by placing them in the appropriate hand-in directory, e.g., `/afs/cs.cmu.edu/academic/class/15410-f17-users/$USER/hw1/$USER.ps` or `/afs/cs.cmu.edu/academic/class/15410-f17-users/$USER/hw1/$USER.pdf`. A plain text file (.text or .txt) is also acceptable, though it must conform to Unix expectations, meaning lines of no more than 120 characters separated by newline characters (note that this is *not* the Windows convention or the MacOS convention). Please avoid creative filenames such as `hw1/my_15-410_homework.PdF`.

1 Chefs (4 pts.)

Jamie, Kelly, and Morgan are chefs in a single kitchen. They cook with knives, bowls, and woks. The maximum number of each item needed by each chef is shown in the following table.

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

Imagine that the kitchen contains 4 knives, 4 bowls, and 4 woks.

Imagine the system is in the state depicted below. List one request which the system should grant right away, and one request which the system should react to by blocking the process making the request. Briefly justify each of your answers. If you can, arrange for both of your answers to involve the same resource request, but made by two different processes.

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

(Continued on next page)

2 "Nemo's Algorithm" (5 pts.)

Consider the following critical-section protocol. This protocol is designed for use by multiple threads. In each thread, `i` is the thread number, ranging from zero through `N`, the number of threads in the system. For the purposes of this question we can assume that `N` is a compile-time constant.

```
      volatile int turn = -1;    // initially, it's nobody's turn
      int entering[N] = {0, };   // per-thread flags (initially all zero)

 1.   int n_entering(void) {
 2.       int n_entering = 0;
 3.       for (int t = 0; t < N; t++)
 4.           n_entering += entering[t];
 5.       assert(n_entering > 0);
 6.       return (n_entering);
 7.   }
 8.
 9.   void lock(void) {
10.       do {
11.           do {
12.               entering[i] = 0;
13.               if (turn == -1)
14.                   turn = i;
15.           } while (turn != i);
16.           entering[i] = 1;
17.       } while (n_entering() != 1);
18.   }
19.
20.   void unlock(void) {
21.       turn = -1;
22.       entering[i] = 0;
23.   }
```

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. You may use more or fewer columns or lines in your trace. If you wish, you may abbreviate `entering[]` as `e[]` and `n_entering()` as `n_e()`.

Execution Trace

time	Thread 0	Thread 1
0		
1		
2		