15-410, Spring 2012, Homework Assignment 1.

Due Tuesday, February 28, 20:59:59

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-s12-users/$USER/hw1/$USER.ps` or `/afs/cs.cmu.edu/academic/class/15410-s12-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 Tape drives (4 pts.)

Consider a system with four processes and seven tape drives. The maximal needs of each process are declared below:

Resource Declarations

Process A	Process B	Process C	Process D
5 tape drives	5 tape drives	3 tape drives	2 tape drives

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.

Who	Max	Has	Room
A	5	1	4
B	5	2	3
C	3	1	2
D	2	1	1
System	7	2	-

2 Bakery Algorithm (6 pts.)

In class we discussed the "Bakery Algorithm" for n-way mutual exclusion, but did not consider how to deal with integer overflow. Below is a version of the algorithm updated to deal with this problem. Note that "for some reason" ticket numbers are fairly small integers (`unsigned short`).

```
// convention 0: arrays are global, scalars are local
// convention 1: for each thread, "i" is that thread's array slot
boolean choosing[MAX_THREADS] = { false, ... };
unsigned short number[MAX_THREADS] = { 0, ... } ;
```

(Continued on next page)

```
while (1) {

  int slot;

  // BEGIN ENTRY PROTOCOL
  // phase 1: pick a number (when we can)
  do {
    unsigned short max = 0;

    choosing[i] = true;

    for (slot = 0; slot < MAX_THREADS; ++slot) {
      if (number[slot] > max) {
        max = number[slot];
      }
    }

    if (max + 1 > 0) // no overflow
      number[i] = max + 1;

    choosing[i] = false;
  } while (number[i] < 1);

  // phase 2: wait while/until we prove our number is best

  for (slot = 0; slot < MAX_THREADS; ++slot) {
    while (choosing[slot]) {
      continue;
    }
    // note: in this language, tuples can be compared with ">"
    while ((number[slot] != 0) && ((number[i], i) > (number[slot], slot))) {
      continue;
    }
  }
  // END ENTRY PROTOCOL

  // BEGIN CRITICAL SECTION
  ...critical section...
  // END CRITICAL SECTION

  // BEGIN EXIT PROTOCOL
  number[i] = 0;
  // END EXIT PROTOCOL

  // BEGIN REMAINDER SECTION
  ...remainder section...
  // END REMAINDER SECTION
}
```

Sadly, this version of the Bakery Algorithm fails to satisfy one (or perhaps more!) of the three critical-section-protocol requirements.

2.1 2 pts

Briefly identify a requirement which is not met and informally suggest why it is not met.

2.2 4 pts

Now your objective is to demonstrate the validity of the claim you made, using the tabular trace format found in the lecture slides.

Please be sure that your trace is clear and conclusive (otherwise your answer will not receive full credit). You may use more or fewer lines or columns than are provided in this sample table. You may change the column headings if you wish.

Execution Trace

time	Thread 0	Thread 1	Thread 2	Thread 3
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				