15-410, Fall 2004, Homework Assignment 1.

Due Wednesday, October 21, 8pm

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, XyWriter, WordStar, etc.). Submit your answers by placing them in the appropriate hand-in directory, e.g., `/afs/cs.cmu.edu/academic/class/15412-f04-users/$USER/hw1/$USER.ps`.

1 First-Come, First-Served? (10 pts.)

While discussing the Bakery Algorithm for N-process mutual exclusion, the textbook claims, on page 197:

> If we wish to show that the progress and bounded-waiting requirements are preserved, and that the algorithm ensures fairness, it is sufficient to observe that the processes enter their critical sections on a first-come, first-served basis.

This claim is true in a sense, but only *almost* true in a strict sense. In other words, it is possible for a process which is "first-come" *in some sense* to enter its critical section after another, "second-come" process. Please use the "thread table format" as exemplfied by the "Failing 'Progress"' slide of the "Synchronization #1" lecture to demonstrate behavior contrary to the textbook's claim.

2 What Could Possibly Go Wrong? (10 pts.)

For Project 2 we asked you to build thread-safe wrappers around `malloc()`, `free()`, and the rest of the standard C-library memory allocation family. Please describe, briefly but in sufficient detail, two horrible things that would happen if you instead used straight "pass-through" wrappers which didn't provide synchronization, e.g.,

```
unsigned char *malloc(unsigned int nbytes) { return _malloc(nbytes); }
```

Part of the point value of the question will depend on how "different" the second horrible thing is from the first. You may describe horrible things based on your understanding of how a C-library memory allocator is typically written and/or examining the source code we released to you in Project 2.

3 Fly Away (15 pts.)

Three people (Delta, Echo, and Foxtrot) at a model-airplane contest are competing to see whose plane, built from a standard kit, can stay aloft the longest. To build and launch a plane, a contestant needs one plane kit, one propellor assembly, and one rubber band. Delta came to the contest with an infinite supply of plane kits, Echo arrived with an infinite supply of propeller kits, and Foxtrot showed up with an infinite number of rubber bands. The contest has a supply table, staffed by an agent ("Alpha"). After each contestant has flown a plane, it is stored in case judges must examine it later to see if any non-standard parts were used. This means that after each flight the contestant must return to the table to obtain supplies for another plane. Each time the judges warehouse a plane, they send a text message to Alpha's phone, at which point it is Alpha's job to reach under the table, remove one each of two different ingredients, selected at random, and place them on the table for use by contestants.

Write a pseudo-code program, using mutexes and condition variables, to synchronize the system. Break your program into four top-level functions (`Alpha()`, etc.), structured as infinite loops, which will be run in separate threads. You may write "helper" functions, and may assume the existence of external functions as follows:

```
/** @brief Effect should be obvious...*/
void build_and_fly_plane(void);

/** @brief Called by Alpha to decide what to put on table */
void pick_ingredients(boolean *supply_kit, boolean *supply_prop, boolean *supply_band);
```

Before you begin, ask yourself which problems you are trying to solve...