Homework 4 — due on Thursday 5 October 2000

Total number of points available on this homework is 100. Full credit is equivalent to 100 points.

1. (10 pts) **Synchronization**. Basic definitions.
   (a) What is mutual exclusion?
   (b) What is an atomic operation?
   (c) What is a semaphore?
   (d) What is busy waiting?
2. (10 pts) **Synchronization.** What advantages does the test&set instruction have over enabling and disabling interrupts? In which circumstances may we still prefer enabling and disabling interupts?
3. (10 pts) **Synchronization.** Suppose you have a process with two threads of execution in it. In general, what code should you place in a critical section? How would you ensure that only one thread at a time was in the critical section?
4. (20 pts) **Synchronization.** Making Water. You need two hydrogen atoms (H), and one oxygen (O) to make water. Write algorithms that generate water as soon as the atoms are available. You do not have to submit actual code (although you may). Pseudocode is sufficient.

   Assume that there are two threads that are making atoms, one for hydrogen and one for oxygen. Each thread has its own queue in which it puts its newly created atoms. A third thread reads atoms off the queues and makes water out of them.
   (a) Solve the problem using semaphores.
   (b) Solve the problem using monitors.
5. (25 pts) **Semaphores.** Suppose a building has a limit on the number of people that may be in it at one time, because of fire code. Suppose this is a very popular place to visit, so the number of people inside must be monitored closely. Further, suppose that this building has more than one entrance and exit. Construct an algorithm that can be used to control a set of turnstiles so that the room can be filled but is never allowed to exceed its legal capacity.

   Assume that each person is a thread that can either request to enter or to leave. The algorithm is a common piece of code and data shared by all person-threads. As before, pseudocode is sufficient.
6. (25 pts) **Monitors.** Consider a family with three children (Bobby, Suzy, and Pat). Each of the three children received one critical piece of ski equipment from their grandmother (Bobby has boots, Suzy has skis, and Pat has poles). However, to go skiing, a child needs a full set of equipment. Each morning the parents put out two distinct pieces of equipment (either skis and boots, skis and poles, or boots and poles), and the child with the remaining piece of equipment can then go skiing.

   Write monitor code for Bobby, Suzy, and Pat that will ensure that one child skis everyday (i.e., there are no deadlocks). You may assume a random number generator that determines which of the two items the parents puts out every morning. Make the Lock_Acquire() and Lock_Release() explicit in your code.

   Assume separate threads for Bobby, Suzy, Pat, and another thread for the parents. Bobby, Suzy, and Pat will use the same code. All four threads will use the same data structures. As before, pseudocode is sufficient.