Exam 1: Processes, CPU Scheduling, & Synchronization Compsci 377

1 hour, closed book, closed notes exam.

1. **Short answer questions. (20pts)**

   Write between one sentence and a paragraph in response to each of the following.

   (a) What differentiates a PCB from a TCB, and what impact does this difference have on context switching?

   (b) To which type of jobs do scheduling policies for interactive systems give priority and why?

   (c) What is the purpose of synchronization?

   (d) What is the advantages does the atomic instruction "test&set" have over atomic loads and stores?

2. **Processes & Threads. (20pts)**

   Write down the sequence of context switches that occur in Nachos if the main thread were to execute the following code. (For example, if the sequence were child2, child1, write "child2 $\rightarrow$ child1".) Assume that there is no time slicing, the CPU scheduler runs the threads in FIFO order, and all threads have the same priority. The "WillJoin" flag tells the thread that its parent will wait for it with a Join().

```
void Thread::SelfTest() {
   Thread *t1 = new Thread(``child 1'', WillJoin);
   Thread *t2 = new Thread(``child 2'', WillJoin);

   t1->Fork((VoidFunctionPtr) &Thread::Yield(), t1);
   t2->Fork((VoidFunctionPtr) &Thread::Yield(), t2);
   t2->Join();
   t1->Join();
}
```

3. **CPU Scheduling. (20pts)**

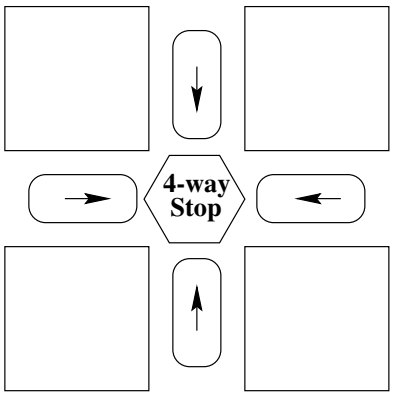
   Consider a variant of Round Robin scheduling in which the usual ready queue (in which PCBs are linked together) is replaced by a queue of pointers to PCBs.

   (a) What is the effect of putting two pointers to a PCB on the queue? (5pts)

   (b) What is one advantage and one disadvantage of this approach? (5pts)

   (c) How could you modify the original Round Robin scheduling algorithm to achieve the same effect without using the queue of pointers? Is your approach better or worse than queues with multiple entries? (10pts)

4. **Synchronization & Deadlock. (40pts)**



Consider this 4-way stop sign, with traffic from all four directions.

(a) Does the rule "yield to the car on the right" prevent deadlock? Why or why not? (10pts)

(b) What is the difference between deadlock and starvation? (5pts)

(c) Instead of the traditional stop sign, you must design an electronic system based on Monitors that allows only one car to enter the intersection at once. Your solution should prevent starvation and deadlock. (25pts)

Below is the code skeleton for one possible solution. You must include the lock->Acquire and lock->Release in each of your Monitor procedures. $d : 0..3$ signifies the four directions from which a car can arrive.

```
class IntersectionMonitor {
  public:
    void Arrive&Stop (d: 0..3);
    void Go (d: 0..3);
  private:
    lock = FREE: Monitor Lock;
    atLight[0..3] = False: Condition Variable;
}
Intersection::Arrive&Stop(d: 0..3){

}
Intersection::Go(d: 0..3){{

}
```