CSc 422 — Homework 3

Due Tuesday, April 9, 2002

This assignment is again worth 40 points. The first four problems are worth 5 points each. The last problem is worth 20 points.

Append a commented listing of your program to your answers to the questions. Also submit your program electronically as described at the end of the assignment. Again follow the programming style described in the handout for Homework 1.

You may as usual discuss the meanings of questions with classmates, but the answers and program you turn in must be yours alone. Explain your answers clearly and succinctly.

1. MPD book, Exercise 5.11.

2. MPD book, Exercise 5.17, part (a).

3. MPD book, Exercise 7.3, part (a).

4. MPD book, Exercise 7.10.

5. (a) Write a program to simulate the Dining Philosophers problem described in Section 4.3 of the text. There should be five processes (threads), one per philosopher, and one "monitor" for the table. You may write your program in MPD, Pthreads, or Java. See the Readers/Writers simulation in the MPD tutorial for an example of how to set up the processes and monitor. Also read Exercise 5.28 of the MPD book.

Your program should have one command-line argument `rounds` that specifies the number of rounds of thinking and eating that each philosopher should execute. The philosophers should eat and think for random amounts of time.

Your program should print a trace of the interesting events in the program. These events include when a process call `getforks`, starts to eat, calls `relforks`, and starts to think. For each event, print a line containing a time stamp (e.g., from the MPD `age()` function), a process id, and a string describing the event. For example, a line of the trace might look like:

```
12345:  Philosopher 4 calls getforks()
```

You may want to have additional command-line arguments, for example to control the eating and thinking intervals and the seed for the random number generator. Please give these default values so we can test your program using just one command-line argument. (If you have additional arguments, describe them in a "Usage: ..." comment at the top of your program.)

Hand in the traces of two interesting (nontrivial) runs of your program.

(b) Your program for part (a) should allow a philosopher to start eating as long as neither neighbor is eating. This can, however, lead to starvation if the processes were to execute forever: One philosopher could be unlucky enough that at all times one or the other of its neighbors is eating, even though each neighbor repeatedly eats and thinks.

Modify your program so that philosophers get to start eating in first-come, first-served order. This means that if one philosopher has to wait to get permission to eat (because a neighbor is eating) and a second philosopher calls `getforks`, the second philosopher *must* wait, even if neither of its neighbors is eating. *Hint*: Use tickets (Section 3.3.2) to order the waiting philosophers in `getforks`.

Turn in a trace that illustrates that your program is indeed enforcing a FCFS order. [For part (b), you must have a working program for full credit, but we will consider pencil-and-paper solutions for partial credit.]

Electronic Turnin. Use `turnin` on Lectura to submit your program. The assignment name is `hw3.philosophers`. The file name for your first program should be `dp.simulation.X`, where `X` is `java`, `c`, or `mpd`. The file name for the second program should be `dp.fair.X`.