UNIVERSITY OF OTAGO EXAMINATIONS 2011

COMPUTER SCIENCE

Paper COSC441

Concurrent Programming

Semester 2

(TIME ALLOWED: THREE HOURS)

This examination consists of 3 pages including this cover page.

Candidates should answer **any 4** questions.

All questions are worth 25 marks, and submarks are shown thus: (5)

No supplementary material is provided for this examination.

Candidates **may not** bring reference books, notes, or other written material.

Candidates **may not** bring calculators into this examination.

TURN OVER

1. **Memory and multicore**

(a) Explain the memory hierarchy and some of its consequences for multicore computers. (10)

(b) Consider the following code fragment:

```
struct Complex {float re, im;};
struct Complex const u = {1.0f, 0.0f};
struct Complex const i = {0.0f, 1.0f};
struct Complex z = u, w;
// in one thread:
z = i;
// in another thread:
w = z;
```

Assuming that loads and stores of float variables are atomic, what are some possible values for w? What does it mean for loads, stores, or any other operation to be atomic? (5)

(c) What POSIX thread feature would you use to manage access to the variable z above? Sketch the code. (5)

(d) What is Transactional Memory? What would code accessing the variable z above look like using Transactional Memory? What are the advantages of Transactional Memory over locking? (5)

2. **Monitors**

(a) Explain monitors. (10)

(b) How would you simulate a monitor using the POSIX threads interface? What guarantees do you get from a compiler-supported monitor abstraction that you do not get from this simulation? (5)

(c) How would you simulate a monitor using Java? What guarantees do you get from a compiler-supported monitor abstraction that you do not get from this simulation? (5)

(d) How would you simulate a monitor using Erlang? What advantage might there be to doing so? (5)

3. **Parallelism, Concurrency, and Distribution**

Compare and contrast parallelism, concurrency, and distribution. What issues does parallelism add to sequential programming? What issues does concurrency add to parallelism? Why is the number of processes or threads part of the semantics of a concurrent program but of only secondary interest in a parallel one? What issues does distribution add to concurrency? How is dealing with failure different in these models? (25)

4. **Shared Memory and Message Passing**

(a) What is shared memory? What is it good for? When is it a good choice for a system? What problems does it create? What is a data race, and how do you prevent one? (10)

(b) What is message passing? What is it good for? When is it a good choice for a system? What problems does it create? What can we say about the order of delivery of messages, why is it hard to say more, and why does it matter? (10)

(c) What is deadlock? How can it arise in a shared memory system? How can it arise in a message passing system? How can it be avoided? (5)

5. **Design and Test**

The day I wrote this exam, several programs on my Mac locked up because the networked file system was unavailable. I was actually writing the exam on another machine, using the Mac as a terminal. The terminal emulator was one of the programs that locked up.

Present the process architecture of a concurrent terminal emulator program, allowing a user to be logged in to several machines at once. You should explain: what are the concurrent activities in the program, what are the resources to be shared by these activities, how sharing could be managed, and how you would test the program. What sensible thing might the terminal emulator be doing that would make it lock up when the file system froze? What would you do so that it didn't *all* lock up? Does it make sense to think of this as an inside-out version of your file server? Why/why not? (25)