UNIVERSITY OF OTAGO EXAMINATIONS 2012

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 **all 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 into this examination.

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 Point3D { double x, y, z; };
struct Point3D const x = {1, 0, 0};
struct Point3D const y = {0, 1, 0};
struct Point3D const z = {0, 0, 1};
struct Point3D w = x;
// In one thread:
w = y;
// In another thread:
w = z;
```

Assuming that loads and stores of `double` variables are atomic, what are some possible states that `w` might end up in? What does it mean for loads, stores, or any other operation to be atomic? (5)

(c) What two POSIX features could you use to manage access to the variable `z` above? Sketch the code for one of them. (5)

(d) What do the Load-Link and Store Conditional instructions do? Why are they *not* immediately useful here? Would the Compare-And-Swap instruction be any better? (5)

2. **Monitors**

A monitor groups together some data, some operations on those data, a lock, and perhaps some condition variables.

(a) Why? (5)

(b) What is a recursive lock, and why might a monitor need to use one? What might happen if a monitor needed to use a recursive lock but used a POSIX default lock? (3)

(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 Java? (4)

(d) How would you simulate a monitor using a message-passing language like Erlang? (3)

(e) Suppose you have an `Account` class in a Java program in which every operation is synchronised, and you need to transfer a sum of money from one Account instance to another. Why isn't Java's automatic locking enough? (4)

(f) What is a total order on locks good for? (2)

(g) What is Transactional Memory and how would it help with the `Account` problem? (4)

3. **Shared Memory and Message Passing**

(a) Explain what shared memory is. Give an example of a programming language that supports shared memory concurrency. What is good about shared memory? What is bad about it? Give an example of a kind of program that might be suitable for shared memory. (8)

(b) Explain what message passing is. Give an example of a programming language that supports message passing concurrency. What is good about message passing? What is bad about it? Give an example of a kind of program that might be suitable for shared memory. (8)

(c) What is deadlock? Use the example of two processes trying to transfer money between the same two `Account`s. What is one way we can avoid deadlock? Can deadlocks happen in a message passing system? Why don't people using NoSQL databases worry about deadlocks? (9)

4. **Design**

(a) What is *flow control* and what is it needed for? (5)

(b) What is the *end to end principle* and why does it matter in system design? (5)

(c) Java has methods to `suspend()` a thread (put it to sleep for a while), `resume()` it (wake it up again) and even `destroy()` it (blast it away completely without any cleanup). Why should you avoid these if you possibly can? (5)

(d) What is a *supervision tree* in Erlang and what is it good for? (5)

(e) What are some issues in testing a concurrent program? (5)