LIT
DEPARTMENT OF
INFORMATION TECHNOLOGY

Semester: Semester 1 (Winter 2016/17)

Date/Time: Friday 9th December 2016, 9:30 AM – 11:30 AM

Programme: Bachelor of Science (Honours) in Computing (Games Design and Development)
Bachelor of Science (Honours) in Computing (Software Development)

Stage: Year 4

Module: CONCURRENT ALGORITHMS

COMP 08034

Time Allowed: 2 hours

Instructions: Attempt any four (4) questions

Additional Attachments: None

External Examiners: Derek O'Reilly
Internal Examiners: Janice O'Connell, Eugene Kenny

Question No. 1 **(25 Marks)**

(a) What are atomic actions in the context of concurrent programs? Distinguish between *fine-grained* and *course-grained* atomic actions. *(10 marks)*

(b) What type of actions are atomic and what problems arise when we rely on them to achieve mutual exclusion? *(15 marks)*

Question No. 2 **(25 Marks)**

(a) What is the difference between *deadlock* and *livelock*? *(10 marks)*

(b) Outline *Peterson*'s algorithm for ensuring mutual exclusion. What are the practical advantages and disadvantages of applying this algorithm. *(15 marks)*

Question No. 3 **(25 Marks)**

(a) What are *semaphores* and show how they can be used to implement condition synchronization. *(10 marks)*

(b) Describe the *Producer-Consumer* problem. Using semaphores, implement a solution to the multiple Producer - multiple Consumer problem using a bounded buffer. *(15 marks)*

Question No. 4 **(25 Marks)**

(a) Explain the differences between *semaphores* and *monitors*. *(10 marks)*

(b) Describe the *Readers and Writers* problem. Outline a *fair* monitor based solution to the Readers and Writers problem. *(15 marks)*

Question No. 5 **(25 Marks)**

(a) What is a `Thread` in Java? Briefly outline the major stages in the life cycle of a thread and explain how the transitions between stages occur. *(10 marks)*

(b) A Java application maintains a queue of deferred tasks. Tasks are represented by instances of classes which implement the `Task` interface, which has a single method `execute()`. *(15 marks)*

Develop a Java `DeferredTaskQueue` class with the following public methods:

- `void DeferredTaskQueue():` (constructor) creates a new `DeferredTaskQueue` object;
- `void append(Task t, long millis)`: add the deferred task `t` to the queue. When `millis` milliseconds have elapsed from the time the task is added to the queue, the task is dequeued and executed; and
- `int length()`: returns the number of deferred tasks currently in the queue.

Your implementation should allow safe concurrent access to the deferred task queue by multiple threads, and should execute the deferred tasks at the specified time (insofar as this is possible given Java's scheduling guarantees). Explain your answer.