DUBLIN CITY UNIVERSITY

SEMESTER ONE REPEAT EXAMINATIONS 2011

MODULE TITLE:	Concurrent Programming
MODULE CODE:	CA463/CA463D
COURSE:	BSc. in Computer Applications (Software Engineering Stream), Study Abroad
YEAR:	4/X
EXAMINERS:	Dr. J. Power Dr. F. Bannister Dr. Martin Crane Ext: 8974
TIME ALLOWED:	2 Hours
INSTRUCTIONS:	Please answer **any 3** questions:

Requirements for this paper
Please tick (X) as appropriate

- ☐ ***Log Table***
- ☐ ***Graph Paper***
- ☐ ***Attached Answer Sheet***
- ☐ ***Statistical Tables***
- ☐ ***Floppy Disk***
- ☐ ***Actuarial Tables***

THE USE OF PROGRAMMABLE OR TEXT STORING CALCULATORS IS EXPRESSLY FORBIDDEN

Please note that where a candidate answers more than the required number of questions, the examiner will mark all questions attempted and then select the highest scoring ones.

PLEASE DO NOT TURN OVER THIS PAGE UNTIL YOU ARE INSTRUCTED TO DO SO

Question 1 **[Total marks: 33]**

1(a) [6 marks]

Show how, using hardware-assisted mutual exclusion, the n process mutual exclusion problem may be solved. Write code to implement this algorithm in SR, carefully explaining your code.

1(b) [12 marks]

Describe the Bakery algorithm for providing mutual exclusion among n processes. Write code to implement the Bakery algorithm in SR. Why is this algorithm not practical?

1(c) [15 marks]

Write code to implement the Bakery algorithm for mutual exclusion of n processes in Java.

--[End of Question 1]--

Question 2 **[Total marks: 33]**

2(a) [5 marks]

What is a monitor?

2(b) [14 marks]

Give a solution to the Dining Philosophers problem using monitors in SR, and prove that deadlock cannot occur. Give in your answer a high level description of the algorithm.

2(c) [14 marks]

The Dining Schoolboys: A class of Schoolboys eats communal dinners from a large pot that can hold M servings of porridge. When a Schoolboy wants to eat, he helps himself from the pot unless it is empty in which case he waits for the pot to be filled. If the pot is empty, the cook refills the pot with M servings. The operations carried out by the cook and the Schoolboys are `fill_pot()` and `get_serving()` respectively. Model the behaviour of the pot using a monitor and write code to implement this monitor in SR.

--[End of Question 2]—

Question 3 **[Total marks: 33]**

3(a) [10 marks]

What are Threads and what support role do they play in Java? How are monitors implemented in Java?

3(b) [23 marks]

Write code to implement the Reader-Preference Readers/Writers Problem with Monitors in Java.

--[End of Question 3]—

Question 4 **[Total marks: 33]**

4(a) [10 marks]

In the context of concurrent programming, what is load balancing? Why is it not practical to find optimal solutions for large load balancing problems?

4(b) [12 marks]

Describe Coffman's load balancing algorithm. How would you classify this algorithm? What assumptions is it based on?

4(c) [11 marks]

Use Coffman's algorithm to schedule the task graph in Figure 1 on to a three processor system.

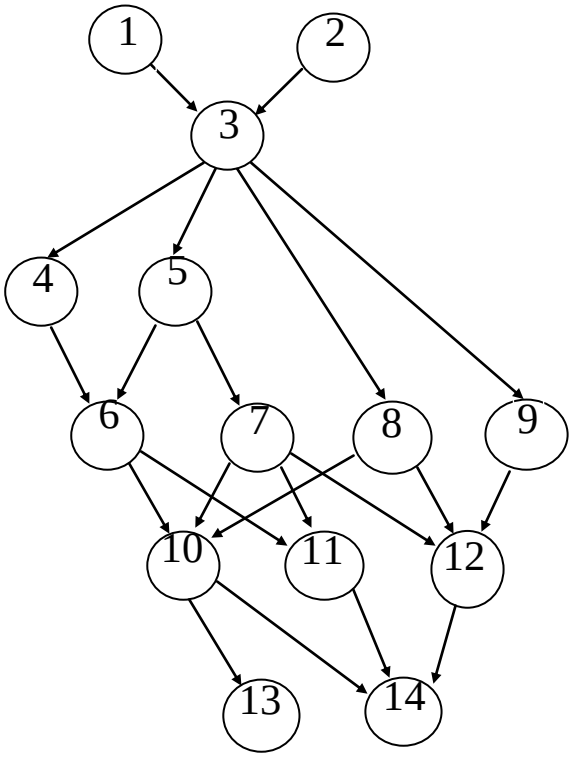

Figure 1

--[End of Question 4]—

Question 5 **[Total marks: 33]**

5(a) [7 marks]

What is the Message Passing Interface (MPI)? List three advantages and three disadvantages of MPI.

5(b) [8 marks]

Write out and explain briefly the arguments of the following MPI commands: MPI_Send, MPI_Recv, MPI_Isend, MPI_Irecv.

5(c) [18 marks]

Write a program in MPI with C bindings that takes data from a master process (process zero), increments it by one and sends it to all of the other processes by sending it in a ring using a *non-blocking Send*. That is, process i should receive the data and send it to process i+1, until the last process is reached. Processes should indicate directly to the user that they have received the data.

Assume that the data sent consists of a single integer and that the master process reads the data from the user.

--[End of Question 5]