DUBLIN CITY UNIVERSITY

SEMESTER ONE EXAMINATIONS 2013

MODULE:	**CA463/D Concurrent Programming**
QUAL:	**B.Sc. in Computer Applications (Software Engineering Stream)** **ECSA/X**
YEAR OF STUDY:	**4/X**
EXAMINERS:	**Dr. J. Power** **Dr. M. Manzke** **Dr. Martin Crane Ext: 8974**
TIME ALLOWED:	**2 Hours**
INSTRUCTIONS:	**Please answer any 3 questions:**

Requirements for this paper
Please mark (X) as appropriate

☐	***Log Tables***
☐	***Graph Paper***
☐	***Dictionaries***
☐	***Statistical Tables***
☐	***Thermodynamic Tables***
☐	***Actuarial Tables***
☐	***MCQ only – Do not publish on Web***

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

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.

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

1(a) [12 marks]

Define Dekker's Algorithm in words and implement it in SR for two processors. Explain clearly all parts of the code.

1(b)

Peterson's algorithm is a variation on Dekker's algorithm whereby each processor uses two variables, `flag` and `turn`. A `flag` value of 1 indicates that the process wants to enter the critical section. The variable `turn` holds the process ID whose turn it is. Entrance to the critical section is granted for process P0 if P1 does not want to enter its critical section or if P1 has given priority to P0 by setting `turn` to 0.

[11 marks]

(i) Implement Peterson's algorithm in Java and briefly compare it with Dekker's algorithm.

[10 marks]

(ii) Show that, in Peterson's algorithm in (i), Mutual Exclusion is satisfied and No Starvation occurs.

--[End of Question 1]--

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

2(a) [4 marks]

What is a semaphore?

2(b) [8 marks]

Write code which solves the Producer-Consumer Problem using circular buffers in SR. Explain clearly all parts of the code.

2(c) [21 marks]

The Hungry Birds Problem: Given that there is a nest with N baby birds and a parent bird. The baby birds eat out of a common dish that initially contains F portions of food. Each baby repeatedly eats one portion of food at a time, sleeps for a while, and then comes back to eat. When the dish becomes empty, the baby bird which empties the dish awakens the parent bird. The parent refills the dish with F portions, then waits for the dish to become empty again. The pattern repeats forever. Represent the birds as processes and develop code that simulates their actions. Using semaphores for synchronization, write code which solves the problem in SR. Assume the existence of `produce`() and `consume`() procedures. Explain clearly all parts of the code. Using the semaphore invariant, prove there can be no starvation.

--[End of Question 2]--

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

3(a) [6 marks]

Explain the difference between Low Level Concurrency Objects and High Level Concurrency Objects in Java. Why would you use the latter rather than the former?

3(b) [12 marks]

Write code for the sleeping barber problem using monitors in SR where the barber and customers are interacting processes, and the barber shop is the monitor in which they interact. You are only required to implement the body of the monitor, not the main code that uses it.

3(c) [15 marks]

Write code for the sleeping barber problem using `lock` and `condition` objects in Java to implement a `Barbershop` class. The barber and customers are interacting processes, and the barber shop is the monitor in which they interact. You are only required to implement the body of the monitor, not the main code that uses it.

--[End of Question 3]—

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

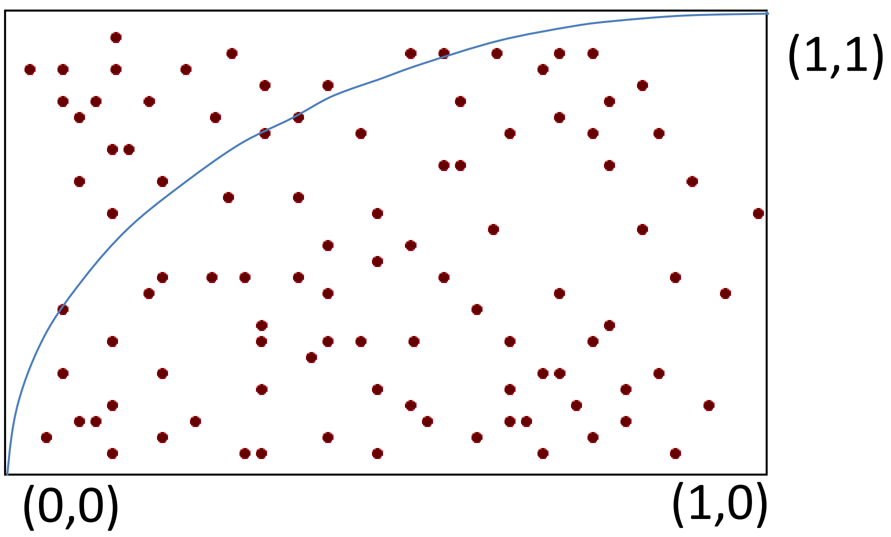

Figure 1 Curve y=sqrt(x) with Random distribution of points

The area of the unit square (shown in Figure 1) is 1 while the area of the under the curve in the range x ε [0,1] of y=sqrt(x) is 0.66667. Write an MPI Program implementing the so-called Monte Carlo technique for computing an approximate value of this area, using the following steps:

- Generate a large number of points at random in the unit square
- Count how many of them fall under the curve
- The fraction of the points within the curve gives an approximation for the area.

--[End of Question 4]