GRIFFITH COLLEGE DUBLIN

QUALITY AND QUALIFICATIONS IRELAND
EXAMINATION

MASTER OF SCIENCE IN BIG DATA
PARALLEL AND DISTRIBUTED PROGRAMMING
Module code: MSCBD-PDP

MASTER OF SCIENCE
PARALLEL AND DISTRIBUTED PROGRAMMING
Module code: MSC-PDP

ONLINE EXAMINATION

Lecturer(s):	**Osama Abushama**
External Examiner(s):	**Dr Mubashir Husain Rehmani**
	Dr Joseph Rafferty
Date: xxth May 2022	**Time:**

THIS PAPER CONSISTS OF FIVE QUESTIONS
FOUR QUESTIONS TO BE ATTEMPTED
ALL QUESTIONS CARRY EQUAL MARKS

HONOUR CODE

By submitting my exam script, I certify that my answers contained in this Examination Script document are entirely composed of my own original work.

During the exam period, which began when I received the exam paper document, I did not work with anyone else on the exam or discuss the examination with anyone else.

I did not access any unauthorised material or copy and hold out as my own any material belonging to or produced by another person.

I understand that failure to adhere to these instructions shall be an Honour Code violation.

Violation of the Honour Code will result in being charged with academic misconduct.

QUESTION 1

(a) Explain what conditions are required for a deadlock to happen, and what measures can be used to resolve a deadlock. Which measure is used in programming?

(10 marks)

(b) Why it is always to use coarse-grained locking versus fine-grained locking.

(5 marks)

(c) What is the main difference between OpenMP and MPI, list two pros and cons of each model?

(10 marks)

Total (25 marks)

QUESTION 2

(a) Write an OpenMP reduction routine that operates on a matrix of size n by n filled by integers, count the numbers of zeros on the diagonal, assume matrix is initialized.

(10 marks)

(b) Explain what is wrong with the following code, what can happen? How to fix it?

```
#pragma omp parallel for
  for (i=1; i < 10; i++)
      {
  factorial[i] = i * factorial[i-1];
      }
```

(5 marks)

(c) What does the OpenMP **nowait** clause do?

(5 marks)

(d) Explain the semantics of the ReentranLock and why it is better than intrinsic locking.

(5 marks)

Total (25 marks)

QUESTION 3

(a) Write a java class that controls an array list buffer of size N for multiple producers and consumers, implement using ReentrantLock. The class has two methods put and get, and a constructor. **Do not write any extra code for threads or main class**.

(15 marks)

(b) Write a Java class that controls access to room with three doors with two methods: enter and leave. Implement this class using semaphores?

(10 marks)

Total (25 marks)

QUESTION 4

(a) Implement a Java program that uses ThreadPool and Callable thread class to find the count of negative and zeros on the diagonal of a matrix of size *1,000,000* x *1,000,000*. Write only the thread class. You are only required to write the callable class.

(10 marks)

(b) What would be simpler solution of part(a) for main program part?

(5 marks)

(c) In OpenMP, what does section accomplish?

(5 marks)

(d) What should never be used with semaphores?

(5 marks)

Total (25 marks)

QUESTION 5

(a) Write an MPI C program to read two Arrays A and B of size N filled with integers and having m machines. The code should place the larger value of each array to a new array C. So for A[i] and B[i], the larger value will be copied to C[i]. Assume arrays A and B have already been initialised.

(15 marks)

(b) For the following three threads, the system is suspectable to deadlock, for each of the scenarios below, determine whether the system is in deadlock and why if the threads are currently on the indicated lines of code.

Scenario A	**Scenario B**	**Scenario C**	**Scenario D**
Thread 1 inside using alpha **Thread 2** blocked on synchronized (alpha) **Thread 3** finished	**Thread 1** finished **Thread 2** blocked on synchronized (beta) **Thread 3** blocked on 2nd synchronized (gamma)	**Thread 1** running synchronized (beta) **Thread 2** blocked on synchronized (gamma) **Thread 3** blocked on 1st synchronized (gamma)	**Thread 1** blocked on synchronized (beta) **Thread 2** finished **Thread 3** blocked on 2nd synchronized (gamma)

In the code below three threads 1, 2, and 3 are trying to acquire locks on objects `alpha`, `beta`, and `gamma`.

Thread 1

```
synchronized (alpha) {
    // using alpha
    // ...
}

synchronized (gamma) {
    synchronized (beta) {
        // using beta & gamma
        // ...
    }
}
// finished
```

Thread 2

```
synchronized (gamma) {
    synchronized (alpha) {
        synchronized (beta) {
            // using alpha, beta, & gamma
            // ...
        }
    }
}
// finished
```

Thread 3

```
synchronized (gamma) {
    synchronized (alpha) {
        // using alpha & gamma
        // ...
    }
}

synchronized (beta) {
    synchronized (gamma) {
        // using beta & gamma
        // ...
    }
}
// finished
```

(10 marks)

Total (25 marks)