GRIFFITH COLLEGE DUBLIN

QUALITY AND QUALIFICATIONS IRELAND
EXAMINATION

POSTGRADUATE DIPLOMA IN SCIENCE IN COMPUTING

PARALLEL AND DISTRIBUTED PROGRAMMING
Module code: PGDC-PDP

MASTER OF SCIENCE IN COMPUTING

PARALLEL AND DISTRIBUTED PROGRAMMING
Module code: MSCC-PDP

MASTER OF SCIENCE IN BIG DATA MANAGEMENT AND ANALYTICS

PARALLEL AND DISTRIBUTED PROGRAMMING
Module code: MSCBD-PDP

ONLINE EXAMINATION

Lecturer(s):	**Osama Abushama**
External Examiner(s):	**Dr Mubashir Husain Rehmani/ Dr Joseph Rafferty**

Date: 2nd June 2021 **Time: 2.15-5.15**

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

HONOUR CODE

By submitting my exam script via Moodle, 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 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 why there is no need to use locks in most MPI programs.

(6 marks)

(b) Explain the differences between OpenMP tasks and OpenMP sections.

(6 marks)

(c) What is the difference between OpenMP single/master and how would the computation be parallelized in MPI?

(6 marks)

(d) Give two other ways to initialize a private variable in OpenMP, with all threads receiving the same value. Can you give scenarios where each of the three strategies would be preferable?

(7 marks)

Total (25 marks)

QUESTION 2

(a) Write an OpenMP reduction routine that operates on an array of non-negative integers, finding the smallest nonzero one. If the array has size zero, or entirely consists of zeros, return -1.

(10 marks)

(b) In the following code, one process sets array A and then uses it to update B; the other process sets array B and then uses it to update A. Argue that this code can deadlock. How could you fix this?

```
#pragma omp parallel shared(a, b, nthreads, locka, lockb)
#pragma omp sections nowait
{
#pragma omp section
{
omp_set_lock(&locka);
    for (i=0; i<N; i++)
    a[i] = ..
    omp_set_lock(&lockb);
    for (i=0; i<N; i++)
    b[i] = .. a[i] ..
    omp_unset_lock(&lockb);
    omp_unset_lock(&locka);
    }
#pragma omp section
    {
omp_set_lock(&lockb);
    for (i=0; i<N; i++)
    b[i] = ...
    omp_set_lock(&locka);
    for (i=0; i<N; i++)
    a[i] = .. b[i] ..
    omp_unset_lock(&locka);
    omp_unset_lock(&lockb);
    }
    } /* end of sections */
} /* end of parallel region */
```

(15 marks)

Total (25 marks)

QUESTION 3

(a) Is the following code correct? Is it efficient? If not, can you improve it?

```
#pragma omp parallel shared(r)
{
int x;
x = f(omp_get_thread_num());
#pragma omp critical
r += f(x);
}
```

(10 marks)

(b) Write a Java class that controls a stack with two methods: Push and Pop. Implement this stack using semaphores?

(15 marks)

Total (25 marks)

QUESTION 4

(a) Implement a Java program that uses ThreadPool and Callable thread class to find the frequency of odd values in a large matrix of size *1000* x *1000* and the location in the last row of matrix. Write the main class and thread class. You are only required to write the callable class, do not write the main.

(15 marks)

(b) In OpenMP, how can one make five threads do completely different code sequences?

(5 marks)

(c) How can you enforce fairness in semaphores?

(5 marks)

Total (25 marks)

QUESTION 5

(a) Write a sequential C program to reads two integer *N* x *N* arrays and perform matrix multiplication of A*B and save results in new matrix C, assume both matrices A and B are diagonal matrices, (i.e. all elements off the diagonal are zeros), this should be taken in consideration to minimize the loops and coding. *N* is a defined constant set to 1000. Assume matrices A and B have already been initialised

(5 marks)

(b) Modify the code using OpenMP directives to parallelize the code using *P* threads, where *P* is also a defined constant and *N* is a multiple of *P*.

(10 marks)

(c) Finally re-write the code to be an MPI program with *P* processes.

(10 marks)

Total (25 marks)