UNIVERSITY OF MORATUWA

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MSc in Computer Science
Semester 2 Examination: 2013/2014

CS5225 PARALLEL AND CONCURRENT PROGRAMMING

Time allowed: 2 Hours August 2013

ADDITIONAL MATERIAL: *None*

INSTRUCTIONS TO CANDIDATES:

1. This paper consists of **6** questions in **6** pages.
2. Answer all questions.
3. Start answering each of the main questions on a new page.
4. The maximum attainable mark for each question is given in brackets.
5. This examination accounts for 40% of the module assessment.
6. This is a closed book examination.

   NB: It is an offence to be in possession of unauthorised material during the examination.
7. Only calculators approved by the Faculty of Engineering are permitted.
8. Assume reasonable values for any data not given in or with the examination paper. Clearly state such assumptions made on the script.
9. In case of any doubt as to the interpretation of the wording of a question, make suitable assumptions and clearly state them on the script.
10. This paper should be answered only in English.

Question 1 (10 marks)

Select the most appropriate answer, and write the corresponding sub question number and the answer number in your answer book. [10 × 1]

(i) Safety property says

A) All processes get a fair share of CPU
B) No process will starve
C) Nothing bad happens ever
D) Something good happens eventually

(ii) "Can" protocol (i.e., use of interrupts) can be used to solve

A) Dining Philosophers problem
B) Mutual exclusion problem
C) Producer and Consumer problem
D) Readers and Writers problem

(iii) Suppose a computer program has a method *M* that cannot be parallelized. *M* accounts for 20% of the program's execution time. Remaining code is parallelized. What is the limit for the overall speedup if it runs on a quad-core processor?

A) 1.6
B) 2.5
C) 4.0
D) 7.5

(iv) In which of the following phase does "already defined tasks and communication structures are evaluated with respect to the performance requirements and implementation costs"?

A) Agglomeration
B) Communication
C) Mapping
D) Partitioning

(v) Which of the following statements are True?

(p) Efficiency of a Parallel Program is ≤ 1
(q) While analysing the performance of a parallel algorithm more readings means better confidence interval
(r) When a solution for a parallel algorithm is scalable, we can solve a larger version of the problem by allocating more resources

A) (*p*) and (*q*) only
B) (*p*) and (*r*) only
C) (*q*) and (*r*) only
D) All three

(vi) Which of the following Solution Patterns for Parallelism can be used to solve the following problem?

```
int[] A = .. int[] B = .. int[] C = ..
for (int i; i < N; i++){
    C[i] = F(A[i], C[i-1])
}
```

A) Divide and Conquer
B) Fork/Join
C) Loop Parallel
D) None of the above

(vii) Which of the following programming approach would you recommend for running a large program on a distributed memory system?

A) MPI | B) MapReduce
C) OpenMP | D) Pthreads

(viii) Which of the following is NOT a step involved while writing a MapReduce job?

A) Write the main( ) function | B) Write the Mapper
C) Write the Master | D) Write the Reducer

(ix) How many threads will be created while executing the following CUDA code?

```
dim3 dimBlock(3, 2)
MyAdd<<<2, dimBlock>>>(a, b);
```

A) 6 | B) 12
C) 36 | D) 64

(x) Which of the following MPI function can be used by a single process to find the minimum value among a large array of numbers?

A) MPI_Allreduce | B) MPI_Gather
C) MPI_Reduce | D) MPI_Scatter

Question 2 (10 marks)

State whether the following statements are TRUE or FALSE. Give one sentence justification for your answer.
Write the corresponding sub question number and the answer in your answer book. [5 × 2]

(i) One way to avoid starvation in Readers and Writers problem is to give priority to writers.

True False

(ii) When latency is not so important and the overall utilization is more crucial, throughput is a good measure of performance.

True False

(iii) In Task Queues, when one thread runs out of work, it goes to another task queue and steals the work.

True False

(iv) MapReduce can be applied to any large data-intensive application.

True False

(v) GPUs are more suitable for SIMD type programs.

True False

Question 3 (5 marks)

Write the most appropriate short answer (word/phrase) for the following questions.
Write the corresponding sub question number and the answer in your answer book. [5 × 1]

(i) ____________________ is a generalized form of Rendezvous.

(ii) A semaphore that DOES NOT guarantee the First In First Out (FIFO) property is called a ____________ semaphore.

(iii) While parallelizing by data, small data units lead to _________________ .

(iv) Two forms of partitioning a problem involve domain decomposition and ______________ decomposition.

(v) In MapReduce, the _____________ function is applied to each entry in the list of data items, which emit (key, value) pairs.

Question 4 (25 marks)

(i) Consider the following program

```
Semaphore s = new Semaphore(5);
int count = 0;

Thread i:
      s.down();
      print("thread "+ i + " work done");
      s.up();
```

(a) Let us assume we run 10 threads using *i* as 1-10. How many threads can be running the *print* statement at the same time? Explain. [3]

(b) Using above as an example, explain how a semaphore works. [4]

(c) Explain what a barrier is, and provide pseudocode to implement a barrier. The barrier does not need to be reusable. [5]

(ii) Suppose you want to parallelize the multiplication of an $m \times n$ matrix and an $n \times 1$ vector.

(a) Write a pseudocode to perform the multiplication parallely using parallel loops. [4]

(b) How much work to be performed? Show steps. [2]

(c) What is the span of the algorithm? Show steps. [2]

(d) How much parallelism is available in the program? Show steps. [2]

(e) Will your algorithm be still useful if m is small and n is large? Briefly Explain. [3]

Question 5 (25 marks)

(i) Compare and contrast Shared Memory and Distributed Memory systems. [5]

(ii) Compare and contrast programming styles involved in CUDA (for GPUs) and MPI. [5]

(iii) Outline an MPI program that can be used to calculate the average of one billion numbers. Once the average is calculated it needs to be informed to all the processes involved in the computation. Use relevant MPI functions that are given in the Appendix. [10]

(iv) Explain how your program works. [5]

Question 6 (25 marks)

(i) Static or dynamic load balancing is essential in most systems to increase the utilization and quality of service. What type of load balancing would you recommend for following problems? Give one sentence justification.

(a) Finding prime numbers among number from 2 to 10^6. [2]

(b) Matrix Vector multiplication. [2]

(c) Indexing web pages found by a web crawler. [2]

(ii) MapReduce is a very simple model but widely used. Give three advantages of using MapReduce to write your parallel program rather than writing it from ground up. [3]

(iii) There are two sets of integers **A** and **B** each provided by a different file with one number for a line with the name *a.set* and *b.set*.

Write pseudocode that shows how you can use MapReduce to calculate the set difference between the two sets. Assume you can get the current file name within the map function by calling *context.getSourceFileName()* method. The answer should provide map and reduce functions. [12]

(iv) Briefly explain in detail how the above map reduce program works when you execute it. [4]

Appendix – MPI Functions

```
. . .
#include <mpi.h>
. . .
int main(int argc, char* argv[]) {
   . . .
   /* No MPI calls before this */
   MPI_Init(&argc, &argv);
   . . .
   MPI_Finalize();
   /* No MPI calls after this */
   . . .
   return 0;
}
```

```
int MPI_Init(int *argc, char **argv)
int MPI_Comm_size(MPI_Comm comm, int *size)
int MPI_Comm_rank(MPI_Comm comm, int *rank)
int MPI_Finalize()
int MPI_Send (void *buf,int count, MPI_Datatype datatype, int dest, int
      tag, MPI_Comm comm)
int MPI_Recv (void *buf,int count, MPI_Datatype datatype, int source, int
      tag, MPI_Comm comm, MPI_Status *status)
int MPI_Reduce(void *sendbuf, void *recvbuf, int count, MPI_Datatype
      datatype, MPI_Op op, int root, MPI_Comm comm)
int MPI_Allgather(void *sendbuf, int sendcount, MPI_Datatype sendtype, void
      *recvbuf, int recvcount, MPI_Datatype recvtype, MPI_Comm comm)
int MPI_Allreduce (void *sendbuf, void *recvbuf, int count, MPI_Datatype
      datatype, MPI_Op op, MPI_Comm comm)
int MPI_Bcast( void *buffer, int count, MPI_Datatype datatype, int root,
      MPI_Comm comm)
int MPI_Gather(void *sendbuf, int sendcnt, MPI_Datatype sendtype, void
      *recvbuf, int recvcnt, MPI_Datatype recvtype, int root,
      MPI_Comm comm)
int MPI_Scatter(void *sendbuf, int sendcnt, MPI_Datatype sendtype, void
      *recvbuf, int recvcnt, MPI_Datatype recvtype, int root,
      MPI_Comm comm)
```

Operation Value	Meaning
MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_SUM	Sum
MPI_PROD	Product
MPI_LAND	Logical and
MPI_BAND	Bitwise and
MPI_LOR	Logical or
MPI_BOR	Bitwise or
MPI_LXOR	Logical exclusive or
MPI_BXOR	Bitwise exclusive or
MPI_MAXLOC	Maximum and location of maximum
MPI_MINLOC	Minimum and location of minimum

-------------------------- END OF THE PAPER --------------------------