Index No:							

UNIVERSITY OF MORATUWA

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B.Sc. Engineering
2013 Intake Semester 7 Examination

CS4532 CONCURRENT PROGRAMMING

Time allowed: 2 Hours June/July 2017

ADDITIONAL MATERIAL: *None*

INSTRUCTIONS TO CANDIDATES:

1. This paper consists of **four (4)** questions in **ten (10)** pages.
2. Answer **All** questions.
3. Answer the questions on the paper itself. **DO NOT** exceed the given space.
4. For MCQ and True/False questions, select the most appropriate answer. No penalty for wrong answers.
5. The maximum attainable mark for each question is given in brackets.
6. This examination accounts for 50% of the module assessment.
7. This is a closed book examination.

   NB: It is an offence to be in possession of unauthorized material during the examination.
8. Only calculators approved by the Faculty of Engineering are permitted.
9. Assume reasonable values for any data not given in or with the examination paper. Clearly state such assumptions made on the script.
10. 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.
11. This paper should be answered only in English.

Q1	**Q2**	**Q3**	**Q4**	**Total**

Question 1 (25 marks)

Circle the correct answer number. [2 × 5 marks]

(i) Which of the following is **not** a contributing factor for the increased adoption of parallel/concurrent programming?

a) Availability of many/multi-core processors
b) Enables better utilization of hardware resources
c) High efficiency of parallel/concurrent programs
d) Is the only way to solve some problems on time

(ii) When a high priority task is indirectly preempted by a low priority task effectively inverting the relative priority of the two tasks, the scenario is called

a) Priority exchange
b) Priority inversion
c) Priority inheritance
d) Priority modification

(iii) A monitor is a module that encapsulates

a) Procedures that operate on shared data structure
b) Shared data structures
c) Synchronization between concurrent procedure invocation
d) All of the above

(iv) Which of the following mutual exclusion algorithm could lead to lower resource utilization under high contention?

a) Centralized algorithm
b) Decentralized algorithm
c) Distributed algorithm
d) Token-based algorithm

(v) MPI style of programming is used in

a) Distributed mutual exclusion problems
b) Distributed memory systems
c) Shared memory systems with uniform memory access
d) Shared memory systems with none uniform memory access

Fill in the blanks. [1 × 5 marks]

(vi) A program with 80% parallelizable code can archive a maximum speed up of ______.

(vii) In *r*-bounded waiting, a process cannot be overtake by more than ___________ times.

(viii) In ____________________ phase (of parallel algorithm design) tasks are combined into larger tasks to improve performance and reduce development cost.

(ix) It is not desirable to have a deeper tree in Divide and Conquer solution pattern as it leads to ____________________ .

(x) Hold and Wait Condition of deadlocks can be avoided by ______________________ ___________________________.

Tick **TRUE** or **FALSE**. Give **one sentence justification**. [2 × 5 marks]

		True	**False**
(xi)	Solutions for concurrency problems must not make any assumptions on time or speed of execution.		
(xii)	Throughput can be increased by reducing latency.		
(xiii)	A semaphore is a shared integer that cannot drop below zero.		
(xiv)	Loop parallel is an example of Parallelizing a problem by Task.		
(xv)	Checkpoint and Rollback prevent the occurrence of a deadlock.		

Question 2 (25 marks)

(i) Using a suitable example(s) briefly explain each of the following terms in the context of concurrent/parallel programming. [2 × 2 marks]

(a) Fairness:

(b) Wait free:

(ii) Consider the following program. Assume initially *turn* = 0.

Process 0

```
flag[0] = 1
while(flag[1]){
  if (turn == 1){
    flag[0] = 0
    while(turn == 1)
    flag[0] = 1
  }
}

… //critical section

turn = 1
flag[0] = 0
```

Process 1

```
flag[1] = 1
while(flag[0]){
  if (turn == 0){
    flag[1] = 0
    while(turn == 0)
    flag[1] = 1
  }
}

… //critical section

turn = 0
flag[1] = 0
```

(a) Explain whether this code satisfy safetiness property or not. [4 marks]

(b) Explain whether this code satisfy liveness property or not. [4 marks]

(c) Is the given code efficient? Explain. [4 marks]

(iii) Consider the following program with two threads.

```
int x = 1;
```

Thread A

```
while(1)
  print "A" + string(x)
  x += 1
```

Thread B

```
while(1)
  X -= 1
  print "B" + string(x)
```

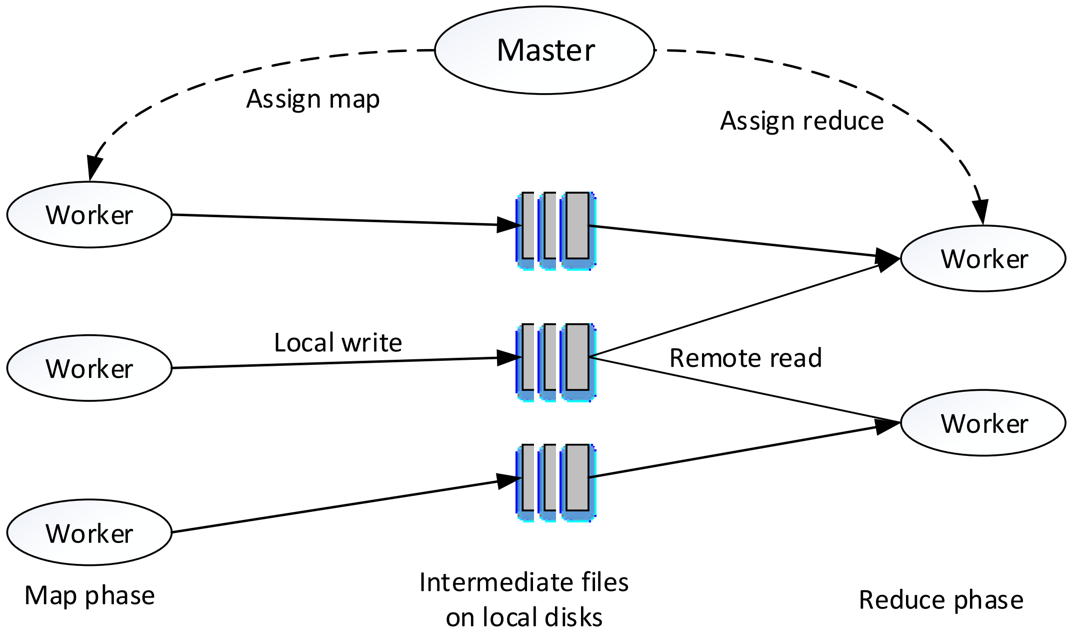
(a) Provide 2 possible outcomes of the above program, except *A0*, *B0*, *A0*, *B0* ... [2 marks]

____ ____ ____ ____ …	____ ____ ____ ____ …

(b) Give a semaphore-based solution to make sure that the program prints only *A0*, *B0*, *A0*, *B0*, … . New code can be introduced only to define the semaphore(s) and to perform *up*() and *down*() operations on the semaphore(s). [7 marks]

Question 3 (25 marks)

Map-Reduce is a good example of producer-consumer problem. As seen in the following figure *Master* assigns *map* and *reduce* tasks to the nodes named *workers*. Workers running *map* function generate intermediate data and store them as files in their local hard disk. Once writing is over, a file is then read by one or more workers running the *reduce* function. Number of workers running *map* and *reduce* functions may vary depending on the problem.



(i) Suppose we want to run map-reduce on a single node with many cores, where each core acts as a worker. Outline a semaphore-based solution for synchronized file access between workers running *map* and *reduce* functions. [10 marks]

(ii) Suppose we want to run map-reduce on a large datacenter, where each worker is a different node. Outline a solution for synchronized file access between workers running *map* and *reduce* functions. [10 marks]

(iii) Once workers running the *map* function write files, workers running *reduce* function need to process them. Propose a solution to balance the workload of workers running the *reduce* function. [5 marks]

Question 4 (25 marks)

Suppose we want to find the total of first 100,000 values of Geometric sequence. Given a number *a* and *r*, the rule is $x_n = ar^{n-1}$. In general, we can write the sequence as:

$$a + ar + ar^2 + ar^3 + ar^4 + ar^5 + \ldots$$

(i) Outline an MPI program (using pseudo code) capable of calculating the sum of first 100,000 values of the sequence, and sending it to all nodes. Indicate MPI functions and key parameters (it is not essential to follow exact function signature). [10 marks]

(ii) Outline a CUDA kernel to generate the sum of first 100,000 values of the sequence. Also, show how you would invoke the kernel. [12 marks]

Kernel launch code:

Kernel code:

(iii) Is the load among MPI nodes and GPU blocks balanced? Discuss. [3 marks]

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