UNIVERSITY OF MORATUWA

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B.Sc. Engineering
2011 Intake Semester 7 Examination

CS4532 CONCURRENT PROGRAMMING

Time allowed: 2 Hours September / October 2015

ADDITIONAL MATERIAL: *None*

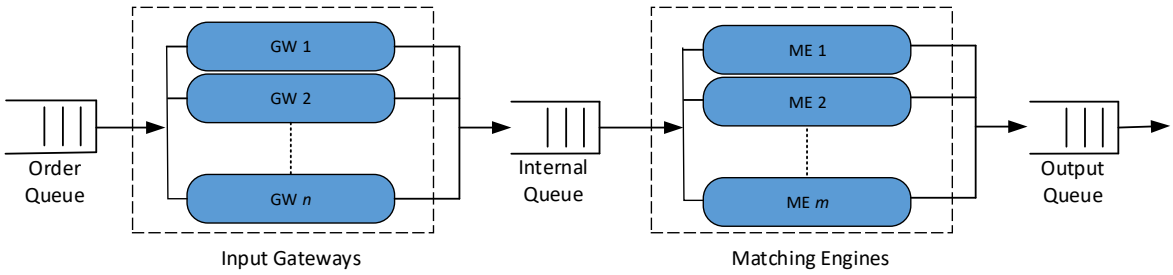
INSTRUCTIONS TO CANDIDATES:

1. This paper consists of **five (5)** questions in **seven (7)** pages.
2. Answer any **four (4)** 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 50% 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 (25 marks)

(i) Following is a simplified setup of a typical Stock Trading System.



The system is expected to handle 75,000 orders/sec and 6,000 concurrent connections. Input Gateway (GW) takes 7 µs to validate an order. Each GW can handle only 1,000 concurrent connections. Matching Engine (ME) is processing heavy; hence, requires 50 µs to match an order. To handle the heavy workload *n* GWs and *m* MEs are to be used.

Orders for the same stock (i.e., same company) need to be processed in First In First Out (FIFO) order. However, orders for independent stocks (i.e., for different companies) may be processed in parallel.

a) How many GWs and MEs are required (i.e., calculate *n* and *m*)? Show key calculation steps. [5]

b) What is the end-to-end latency (from Order queue to Output queue) to process a single order? Assume network latency is 3 µs and order result can be directly send to the user without going through another GW. [2]

c) Discuss what changes are required to the above system to handle burst (i.e. bunch) arrival of orders, where their inter-arrival times may be less than 40 µs. [5]

d) What Load Balancing technique do you recommend for the following 2 queues? Give a brief justification for each.

   i. Order queue [3]

   ii. Internal queue [3]

e) "*We can further reduce the processing time of each Matching Engine by utilizing more and more CPU cores*".

   Do you agree or disagree with this statement? Justify.

   Hint: Consider nature of the workload and Amdahl's law. [3]

f) What do you propose to use for the Stock Trading System, "a single server with a large number of cores" or "a cluster of nodes"?

   Suggest your recommendation while considering advantages and disadvantages of each approach. State any assumptions. [4]

Question 2 (25 marks)

(i) Consider the following program with 2 threads.

```
int i;
```

Thread 1

```
i = 0;
i++;
printf ("%d ", i);
```

Thread 2

```
i = 0;
i--;
printf ("%d ", i);
```

a) Provide 4 possible outcomes of the above program. [4]

b) Following solution based on a Condition Variable is suggested as a solution to make sure that the above program generates only "1, -1" as the output.

```
int i;
CV;
Lock l;
Boolean done = False;
```

Thread 1

```
i = 0;
i++;
printf ("%d ", i);
l.lock();
    done = True;
l.unlock();
cv.notify();
```

Thread 2

```
l.lock();
if( !done )
    cv.wait()
l.unlock()
i = 0;
i--;
printf ("%d ", i);
```

Briefly explain the functionality of this program and discuss whether the given code satisfy safety and liveness properties while attempting to print "1, -1". [10]

c) Give a semaphore-based solution to make sure the above program generates only "1, -1" as the output. [5]

(ii) Recommend a suitable solution pattern to parallelize the following code snippets.

Provide a suitable justification for each case. State any assumptions.

i.

```
for(k = 1, k < 500; k++){
    x[k] = y[k – 1] + 1;
    y[k] = z[k – 1] + 2;
}
```

[3]

ii.

```
for(k = 1, k < 500; k++){
    x[k] = y[k + 1] + 1;
    y[k] = x[k + 1] + 2;
}
```

[3]

Question 3 (25 marks)

(i) Give an example for each of the following cases where the particular implementation of readers and writers solution becomes useful.

a) Readers-writers solution that gives priority to readers. [3]

b) Readers-writers solution that give priority to writers. [3]

(ii) A group of students are studying for CS 4532 exam. The students can study only while eating pizza. Each student executes the following loop:

```
while (true) {
    pick up a slice of pizza;
    study while eating the pizza
}
```

If a student finds that the pizza is gone, the student goes to sleep until another pizza arrives. The first student to discover that the group is out of pizza calls *Kamal's Pizza* to order another pizza before going to sleep. Each pizza has *s* slices. Once Kamal delivers pizza, he wake up all the students in the group. Then the students pick up a slice of pizza and go back to studying, and the process continues.

Write code to synchronize the student threads and the Kamal's pizza delivery thread.

Your solution should avoid deadlock and call Kamal's Pizza (i.e., wake up the delivery thread) exactly once each time a pizza is exhausted. No slice/piece of pizza may be consumed by more than one student. Comment your code.

Hint: Think about the implementations of readers-writers and barrier solutions. [19]

Question 4 (25 marks)

(i) Consider the following sequential computation. This computation is a 2-dimensional Jacobi relaxation on an $N \times N$ grid, where N is unknown. For simplicity, you can assume N is evenly divisible by 32.

```
Jacobi(float *a, float *b, int N){
    for (i = 1; i < N -1; i++){
        for (j = 1; j < N - 1; j++){
            a[i][j]=0.8*(b[i - 1][j]+b[i + 1][j] +b[i][j - 1] + b[i][j+1]);
        }
    }
}
```

a) Draw the stencil pattern that illustrates the above computation. Only consider 2D array *b*. [4]

b) Assume that you are parallelizing the 2 loops, and each thread computes just one element. Your goal is to parallelize it such that you use a cyclic distribution in both the *x* dimension and *y* dimension of threads.

Outline a CUDA kernel for the Jacobi calculation. Following code is to be used to launch the CUDA kernel.

```
dim3 dimGrid(N/32,N/32);
dim3 dimBlock(32,32);
Jacobi_GPU<<<dimGrid,dimBlock>>>(A,B,N);
```

[15]

(ii) Are the use of following 2 methods free from deadlocks? Discuss. [6]

```
public void method1() {
    synchronized (String.class) {
        System.out.println("Aquired lock on String object");

        synchronized (Integer.class) {
            System.out.println("Aquired lock on Integer object");
        }
    }
}

public void method2() {
    synchronized (Integer.class) {
        System.out.println("Aquired lock on Integer object");

        synchronized (String.class) {
            System.out.println("Aquired lock on String object");
        }
    }
}
```

Question 5 (25 marks)

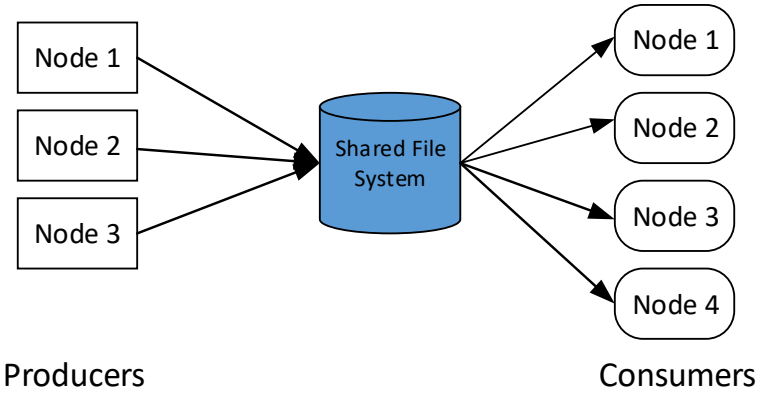
(i) The Root Mean Square (RMS) is one of the several kinds of averages. It is often used in many engineering and statistical applications, e.g., in electrical engineering. RMS of *n* real numbers x_1, x_2, x_3, ... x_n can be calculated as follows:

$$RMS(X) = \sqrt{\frac{\sum_{i \in n} x_i^2}{n}} = \sqrt{\frac{x_1^2 + x_2^2 + x_3^2 + \ldots + x_n^2}{n}}$$

Outline an MPI program (using pseudo code) that can be used to calculate the RMS of one million real numbers. Once the calculation is complete, mean should be stored on a variable at process 0.

Use relevant MPI functions that are given in the Appendix. Note that it is impractical to create one million concurrent processes/threads. [15]

(ii) A producer-consumer implementation is to use a shared file system to send the workload generated by a producer to a consumer (e.g., in web crawling). Multiple producers and consumers are to be used as follows.



The system must ensure that once a producer *x* writes to a file *f* then only a consumer *y* should read from *f*.

a) In this problem, what is the shared resource and who are the accesses trying to access that resource? [2]

b) Outline a solution to this problem assuming this is a shared memory problem. [4]

c) How would you solve this distributed mutual exclusion problem, where you do not have access to a shared memory? [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 ----------