UNIVERSITY OF MORATUWA

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B.Sc. Engineering
2011 Intake Semester 8 Examination

CS4532 CONCURRENT PROGRAMMING

Time allowed: 2 Hours February/March 2016

ADDITIONAL MATERIAL: *None*

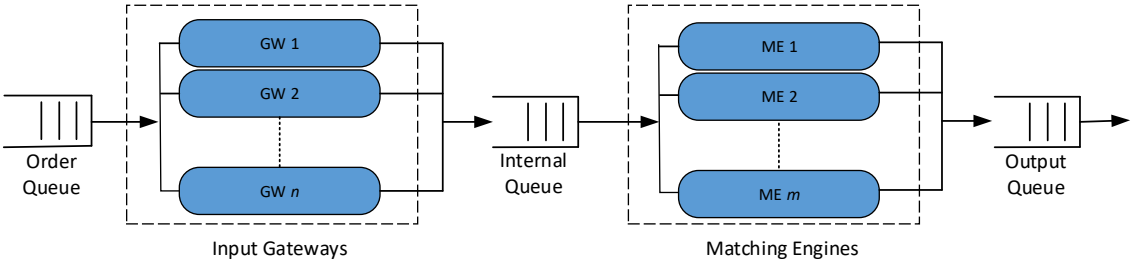
INSTRUCTIONS TO CANDIDATES:

1. This paper consists of **five (5)** questions in **six (6)** 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)

Following is a simplified setup of a typical Stock Trading System.



The system is expected to handle 50,000 orders/Sec and 5,000 concurrent connections. Input Gateway (GW) takes 8 µs to validate an order. Each GW can handle only 1,000 concurrent connections. Matching Engine (ME) is processing heavy; hence, requires 60 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.

(i) How would you apply the four steps of Parallel Algorithm Design to solve this problem?

You are expected to explain how Partitioning, Communication, Agglomeration, and Mapping steps are applied in the context of this problem. [8]

(ii) How many GWs and MEs are required (i.e., calculate *n* and *m*)? Show key calculation steps. [4]

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

(iv) Discuss what changes are required to the above system to handle burst (i.e., bunch) arrival of orders, where sometimes their inter-arrival times may be less than 30 µs. [5]

(v) "*Based on past statistics it is known that within a unit time, 55% of the buy and sell orders are correlated where they are for the same stock. Hence, we can further reduce the processing time of each Matching Engine by utilizing GPUs*".

Do you agree or disagree with this statement? Justify.

Hint: Consider nature of the workload and Amdahl's law. [6]

Question 2 (25 marks)

(i) Consider the following program with 2 threads.

```
int i;
i = rand()% 2 - 1
```

Thread 1

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

Thread 2

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

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

b) Give a semaphore-based solution to make sure the above program generates only "0, -1" as the output. [6]

c) It is possible to use a conditional variable to provide the same solution? [2]

(ii) A distributed conference app provides a shared whiteboard/wall which can be accessed via smart phones and tablets. Each member of the conference has a replica of the whiteboard, which could be used to post one's own messages, photos, and events, as well as see others posts related to the conference.

d) Propose a suitable solution that allows each conference participant to achieve mutually exclusive access to the whiteboard, prior to the propagation of updates to all the conference participants. [8]

e) Justify to what extent your proposed solution provides mutual exclusion and free from issues like deadlocks and starvation. [5]

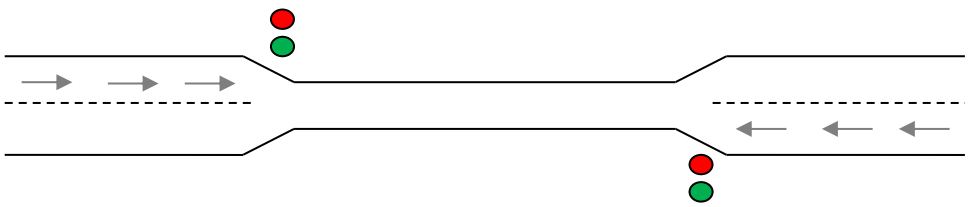
Question 3 (25 marks)

(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 writers. [3]

b) Readers-writers solution that occasionally allows either readers or writers than giving priority to a specific group. [3]

(ii) Following is a two-lane east-west road that passes through a one-lane tunnel. A car can safely enter the tunnel, if and only if there are no oncoming cars in the tunnel. To prevent accidents, sensors installed at each end of the tunnel notify a controller computer when cars arrive or depart the tunnel in either direction. The controller uses the sensor input to control signal lights at either end of the tunnel.



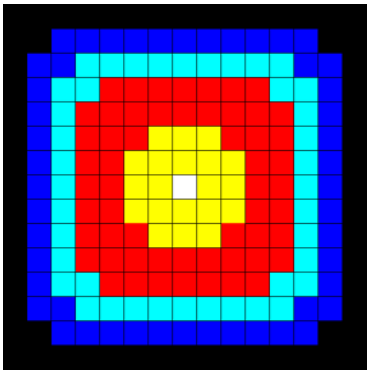
Give an implement of the controller program using mutexes and condition variables.

You may assume that each car is represented by a thread that calls *Arrive*() and *Depart*() functions in the controller, passing an argument indicating the direction of travel. You may also assume that *Arrive*() can safely stop the arriving car by changing the correct signal light to red and blocking the calling thread. Your solution should correctly handle rush hour, during which most cars approach the tunnel from the same direction.

Hint: Your solution should not rely on strict alternation. [19]

Question 4 (25 marks)

Suppose you are planning to simulate the heat transfer on a 2D plate as shown in the following diagram.



The heat source is at the centre of the 2D plate. Corresponding heat transfer/conduction equation is as follows:

$$T_{x,y} = T_{x,y} + C_x(T_{x+1,y} + T_{x-1,y} - 2T_{x,y}) + C_y(T_{x,y+1} + T_{x,y-1} - 2T_{x,y})$$

Where $T_{x,y}$ is the temperature at location (x, y), i.e., $T[x, y]$ in 2D array T. C_x and C_y are constants.

This equation is applied for each and every point within the 2D surface. Once the temperature of all the points at time t is calculated, those temperatures are then used to calculate the temperature at time $t + \delta$ by re-applying the equation to all the points. The process continues until the end of simulation time is reached (i.e., $m\delta$ is reached, where $m > 1$ is the number of simulation rounds).

(i) Draw the stencil pattern that illustrates the above computation. [4]

(ii) Recommend a suitable solution pattern to parallelize the computation of heat transfer. Justify. [4]

(iii) Outline a CUDA kernel for the heat transfer equation. Following code is to be used to launch the CUDA kernel.

```
dim3 dimGrid(N/32,N/32);
dim3 dimBlock(32,32);
HeatTransfer_GPU<<<dimGrid,dimBlock>>>(T,N);
```

Where N is the size of the 2D plate. [14]

(iv) How do you ensure that after applying the equation for a given round, final result after m rounds remains accurate? Discuss. [3]

Question 5 (25 marks)

(i) Suppose you are given one million random integers. You need to find the sum of all the integers which are prime, i.e.,

$$\text{Sum of All Primes} = \sum_{\text{If } x_i \text{ is prime}} x_i$$

Numbers are not in any particular order and are stored in a file accessible via a shared file system.

a) Outline an MPI program (using pseudo code) that can be used to calculate the sum of all given prime numbers. Once the calculation is complete, result 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. [11]

b) Comment on the performance of your program and its ability to full utilize all the computational nodes available in the MPI cluster. [4]

(ii) The Dining Philosophers have worked up a solution to avoid deadlock, with a little help from a consultant. Before eating, each philosopher will flip a coin to decide whether to pick up the left fork or the right fork first. If the second fork is taken, the philosopher will put the first fork down, then flip the coin again.

a) Is this solution deadlock-free? Discuss. [4]

b) Does it guarantee that philosophers will not starve? Discuss. [3]

c) Does it guarantee that philosophers cannot form a pact/agreement to prevent one or more philosophers from eating (as in Tanenbaum's solution)? Discuss. [3]

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 ----------