University of Minnesota
Department of Computer Science
CSci 5103 - Spring 2018 (Instructor: Tripathi)
Midterm Exam 1 — Date: February 28, 2018 (1:00 pm – 2:15 pm)
CLOSED BOOK/NOTES
(Time: 75 minutes) Total Points – 100

STUDENT NAME:
STUDENT ID:

Problem 1	Problem 2	Problem 3	Problem 4	Problem 5	Total Score
20	20	20	20	20	100

Problem 1: (20 points)
(a) (4 points) Even when a process is swapped out to disk, some information has to be maintained for it in the process table which is always kept in the primary memory by the kernel. Identify **at least four items** of information that must always be maintained in the primary memory.

(b) (2 points) In many of the modern shared-memory multiprocessor systems, processes contend for a critical section using a spinlock, which is implemented using an instruction such as *test-and-set*. Under what conditions this approach of using spinlock-based critical sections justifiable? Briefly justify your answer.

(c) (2 points) Identify **at least two** important differences between a counting semaphore and a condition variable in a monitor.

(d) (3 points) What is the effect of **decreasing** the time quantum of a round-robin scheduler on the following performance measures? Will the values of the following measures <u>increase</u> or <u>decrease</u>?

- Turnaround time of long jobs?

- Throughput of short jobs?

- Turnaround time of short jobs?

(e) (2 points) Does kernel code make system calls using trap instructions? Answer YES or NO.

(f) (4 points) In Lamport's Bakery Algorithm for N processes, can the sequence-number part in a ticket become arbitrarily large when the system has been running for a long time? (Give a brief justification.)

What is the upper-bound on the difference between the sequence-numbers of two tickets held by any two processes competing for critical section? Answer either YES or NO. (Give a brief justification.)

(g) (3 points) Identify **at least three** conditions under which operating systems may boost the priority of a process temporarily.

Problem 2 (20 points): Consider a system with two processors. There are five jobs – A, B, C, D, and E – waiting in the ready queue to be processed. Their respective service times, as they appear in the queue (starting with the job at the head of the queue) are 5, 4, 3, 2, and 1 seconds. Assume that the jobs arrived at the same time, and a job can be executed on any of the available processors.

(a) (8 points) For the following two scheduling policies, draw the timeline diagrams for each of the two processors, showing the jobs being executed on these two processors.

- FCFS
- Shortest Job First

(b) (8 points) Determine the average turnaround time for each of the two scheduling policies in part (a) above.

(c) (4 points) Compute the average waiting time for the two cases in part (a).

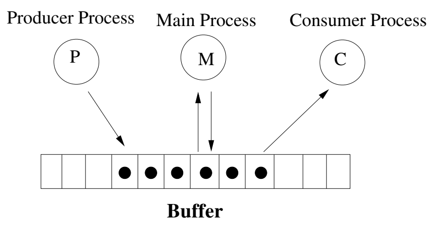
continue with your answer of Problem 2

Problem 3 (20 points): Consider a real-time system consisting of two periodic tasks. Assume that both these tasks require only the CPU and do not perform any I/O. Task A has period of 50 seconds and requires 25 seconds of CPU time. Task B has period of 75 seconds and requires 30 seconds of CPU time. For each task, its period is also the deadline.

1. Is the condition for the rate monotonic (RM) scheduling satisfied? Is a schedule possible using RM-based static priorities? If yes, show a schedule, otherwise show a case where a task deadline is missed with RM based scheduling. (Note: For any calculations, $\sqrt{2} = 1.414$)
2. Now, suppose that we upgrade the CPU of this system with a CPU which is 25% more powerful than the original one, i.e. the power of the new CPU is 1.25 times that of the original one. Is the RM scheduling condition satisfied? Is it possible to schedule these tasks using the rate monotonic (RM) scheme? If yes, give a schedule.

Problem 4: (20 points)

Consider a set of three communicating processes consisting of a main process M, a producer process P, and a consumer process C. These processes execute concurrently and asynchronously such that P produces some data item and passes it to process M, which performs some operation on each data item and passes it to the consumer process C which prints each processed data item. **Only one buffer of size N is given**, which is implemented using an array of size N (buffer indices are 0 through N-1).



Using **semaphores** write code for these three processes to properly synchronize their access to the shared buffer.

Process P repeatedly puts a data item in the buffer if there is space available.

Process M repeatedly gets the next available data item to be processed. It gets the item and after processing it puts it in at the same place in the buffer from where it obtained the item.

Process C repeatedly removes the next processed item from the buffer for printing. It removes the item from the buffer and prints it.

continue answer for problem 4...

Problem 5 (20 points)

In this problem write a Hoare monitor to properly synchronize the use of a shared restroom facility. When a woman is using the facility, other women can enter and use the facility, but no men can enter it, and vice versa.

This monitor will have the following four methods: *woman_wants_to_enter*, *woman_leaves*, *man_wants_to_enter*, *man_leaves*, The monitor will keep track of the status of the restroom such as *empty*, *women-using*, *men-using*, and appropriately block a user from using the facility.

The protocol for using the facility by a man would look as follows:

```
 RestroomMonitor.man_wants_to_enter();
   Enter and use the restroom facility;
   Leave the facility;
RestroomMonitor.man_leaves();
```

Write code for only two of these four methods: *woman_wants_to_enter* **and** *man_leaves*
.

continue answer to problem 5...