University of Minnesota
Department of Computer Science & Engineering
CSci 5103 - Fall 2018 (Instructor: Tripathi)
Midterm Exam 1 — Date: October 18, 2018 (1:00 – 2:15 pm)
(Time: 75 minutes) Total Points – 100
This exam contains five questions.
CLOSED BOOK/CLOSED NOTES – NO Laptops or Cell Phones
Please write your answer in the space provided with each question.

STUDENT NAME:
STUDENT ID:

<table>
<thead>
<tr>
<th>Problem 1</th>
<th>Problem 2</th>
<th>Problem 3</th>
<th>Problem 4</th>
<th>Problem 5</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>20</td>
<td>12</td>
<td>18</td>
<td>25</td>
<td>100</td>
</tr>
</tbody>
</table>
Problem 1 (25 points):
(a) (2 points) Select the correct answer: a process making a system call passes the parameters of the call to the kernel code by writing them to:

(1) Kernel’s stack? YES NO
(2) Process’s stack? YES NO
(3) CPU register? YES NO
(4) Kernel memory? YES NO

(b) (4 points) Even when a process is swapped out to disk, some process management information related to it 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 by the kernel in the primary memory.
(c) (2 points) In this problem you have to compare the performance of two scheduling disciplines — FCFS and Round-Robin. Given is a fixed set of jobs in which all jobs have the same CPU service-time requirements. Assume that the time-quantum used in the Round-Robin scheme is much smaller (say, by at least one or two orders of magnitude) than the CPU requirement of a job.

Which of these two disciplines will result in higher average turn-around time?
Select Answer: FCFS Round-Robin

Which of these two disciplines will result in higher variance for the turn-around time?
Select Answer: FCFS Round-Robin

(d) (2 points) Consider a multithreaded process with threads implemented by a user-level library. Select the correct answer
Is there a separate kernel-level stack for each thread? NO YES
Is there a separate user-level stack for each thread? NO YES

(e) (2 points) Consider a multiprogrammed computer system with four processes. Suppose that each process spends $x\%$ of its time in performing I/O. What is the probability of CPU being idle?
In this problem you are to compare the performance of a web-server implemented as a single-threaded server or a multithreaded server. (Assume that threads are supported by the kernel.) Suppose that it takes 15 milliseconds to get a request for work, despatch it, and do the rest of the necessary processing, assuming that the data needed are in the main memory. If a disk operation is needed, as is the case one-third of the time, an additional 75 millisecond is required, during which time the thread sleeps.

**Part A:** (4 points) How many requests per second can this server handle if it is single-threaded?

**Part B:** (4 points) How many requests per second can this server handle if it is multi-threaded?
(g) (5 points): For the reader/writer synchronization problem, the following code is a slight modification of the code presented in the lectures. Is this solution correct? Justify your answer.

Shared data:
   int readCount = 0;
   semaphore mutex = 1;
   semaphore wrex = 1;

WRITER Process::
   wait ( wrex );
   PERFORM WRITE
   signal ( wrex );

READER Process::
   wait ( mutex );
   readCount++;
   if (readCount == 1) then {
      signal ( mutex ); //release critical section
      wait ( wrex );
   }
   else signal ( mutex );
   PERFORM READ
   wait ( mutex );
   readCount--;
   if (readCount == 0) then {
      signal ( wrex );
   }
   signal ( mutex );
Problem 2: (20 points)

Five batch jobs $A$, $B$, $C$, $D$, and $E$ arrive at time 0 in this given order. They have estimated running times of 10, 8, 12, 4, and 6 seconds. Their externally determined priorities are 5, 1, 3, 4, and 2, respectively, with 5 being the highest priority. For each of the following four scheduling algorithms, determine the average turn-around time and average waiting time. Ignore process context-switch overhead.

(a) Round-Robin with processor-sharing, i.e. each jobs gets its fair share of the CPU.
(b) Priority based scheduling
(c) Shortest Job First
(d) FCFS
Continue answer for Problem 2
Problem 3 (12 points): Write a monitor called ExchangeBox that will be used by two asynchronous processes to swap two integer values.

Each process would call the monitor procedure Swap(integer i).

Whichever process calls this procedure first would be blocked, waiting for the second process to make this call. When the other process calls Swap, their integer parameters would be swapped and then respectively returned as the result of the call.

For example, suppose that process A calls Swap(10) and process B calls Swap(20). Process A will return with result value 20, and B with value 10.
Continue answer for Problem 3
Problem 4 (18 points): Consider a real-time system with two periodic tasks. Task A has period of 50 seconds and it requires 15 seconds of service time. Task B has period of 75 seconds and it requires 40 seconds of service. At time 0, both these tasks arrive, and then they repeat according to their periods given above.

1. (4 points) Is the RMS condition satisfied for this system?

2. (4 points) Is a schedule possible using the Rate Monotonic scheduling?

3. (4 points) Is a schedule possible using the Earliest Deadline First scheduling? If so, show the execution of these tasks from time 0 to 150.

4. (6 points) Is a schedule possible using the Least Slack-Time First scheduling scheme? If so, show the schedule from time 0 to 150, assuming that the scheduler computes the slack values for the two tasks at every 1 second interval to make the scheduling decision.
Continue answer for Problem 4
Problem 5: (25 points) Using semaphores, write the synchronization code for a system with two producer processes and one consumer process. This system contains one buffer with $2N$ slots for communicating items from the producers to the consumer. The producer processes jointly produce an item consisting of two parts. Each part of an item occupies one buffer slot, and the two parts of an item occupy consecutive slots in the buffer. The two parts may be stored in any order in the buffer, but they must be placed consecutively. The consumer process always removes a complete item from the buffer.

Write the pseudo-code for the producer and the consumer processes, using semaphores to synchronize them.
Continue answer for Problem 5
Blank page, if you need one for writing answers or scratch work.