Carleton UNIVERSITY

Sample Solution to Fall 2017 MIDTERM
EXAMINATION

DURATION: 1.5 HOURS **No. Of Students: 138**

Department Name & Course Number: Systems and Computer Engineering SYSC 4001A

Course Instructor: Thomas Kunz

AUTHORIZED MEMORANDA:

William Stallings, *Operating Systems: Internals and Design Principles*, 9th edition, Pearson 2018, ISBN-9780134670959 (as physical book, no ebook) or earlier versions of that same book. Also, a non-programmable calculator is allowed.

Students MUST count the number of pages in this examination question paper before beginning to write, and report any discrepancy to a proctor. This question paper has 4 pages + cover page = 6 pages in all.

This examination question paper MAY NOT be taken from the examination room.

In addition to this question paper, students require: an examination booklet: NO
Scantron Sheet: NO

Name: _____

Student Number: _____

Question 1: _____ /10

Question 2: _____ /10

Question 3: _____ /10

Question 4: _____ /10

Total: _____/40

Exam questions will not be explained, and no hints will be given. If you think that something is unclear or ambiguous, make a reasonable assumption (one that does not contradict the question), write it at the start of the solution, and answer the question. Do not ask questions unless you believe you have found a mistake in the exam paper. If there is a mistake, the correction will be announced to the entire class. If there is no mistake, this will be confirmed, but no additional explanation of the question will be provided.

Question 1: Processes and Threats (10 marks)

a. Two advantages of using multiple thread within a process are that (1) less work is involved in creating a new thread within the existing process than creating a new process and (2) communication among threads within the same process is simplified. Is it also the case that a mode switch between two threads within the same process involves less work than a mode switch between two threads in different processes?

Answer (3 marks):
Yes, because more state information must be saved to switch from one process to another.

b. In the discussion of ULT versus KLT, it was pointed out that a disadvantage of ULTs is that when a ULT executes a system call, not only is that thread blocked, but also all of the threads within the process are blocked. Why is that so?

Answer (3 marks):
Because, with ULTs, the thread structure of a process is not visible to the operating system, which only schedules on the basis of processes.

c. Consider an environment in which there is a one-to-one mapping between user-level threads and kernel-level threads that allows one or more threads within a process to issue blocking system class while other threads continue to run. Explain why this model can make multithreaded programs run faster than their single-threaded counterpart on a uniprocessor computer.

Answer (4 marks):
The issue here is that a machine spends a considerable amount of its waking hours waiting for I/O to complete. In a multithreaded program, one KLT can make the blocking system call, while the other KLTs can continue to run. On uniprocessors, a process that would otherwise have to block for all these calls can continue to run its other threads**.**

Question 2: Concurrency: Mutual Exclusion and Synchronization (10 marks)

A file is to be shared among different threads, each of which has its own unique ID number (for simplicity, assume that a thread i's ID number is i). The file can be accessed by several threads at the same time as long as the sum of the thread ID numbers is less than a constant `k`. Write pseudocode for Thread(i) that coordinates access to the file using semaphores while ensuring the above synchronization requirement.

Note: it is tempting to redefine the semaphore operations, which would potentially make this question quite simple to solve. However, that is not how things work in real life: semaphores have a fixed and well-defined set of operations and you should limit yourself to using only those.

Solution (10 marks):

```
int current_sum = 0;
int threads_blocked = 0;
semaphore lock = 1;
semaphore IDsum = 0;

Thread i { // i is the ID of the thread
      // check whether thread i can be admitted
      lock.semWait();
      while (i + current_sum !< k) {
               threads_blocked += 1; // count thread as being blocked
               lock.semSignal(); // leave critical section
               IDsum.semWait();  // block yourself
               lock.semWait();   // re-enter critical section
      }
      current_sum += i;
      lock.semSignal();

      Access File;

      lock.semWait();
      current_sum -= i;
      for (count = 0; count < threads_blocked; count++) {
               IDsum.semSignal(); // wake up all potentially pending threads
      }
      threads_blocked = 0;
      lock.semSignal();
}
```

Key points for the solution (which is a variant of Question 5.16 in the textbook):

1. `current_sum` and `threads_blocked` are shared variables for all threads, so protect access to them via the semaphore `lock` (initialized to 1). `current_sum` keeps track of the sum total of IDs that have been allowed to proceed, `threads_blocked` explicitly counts how many threads are currently blocked from accessing the file.
2. A thread may be prevented from making progress because the inequality `current_sum + i < k` is not true. We use a second semaphore `IDsum` to block such threads. However, if the blocking occurs within the first critical section, all other threads would be prevented from accessing the file or increasing `current_sum` when they are done with the file access. So in the loop body we first "leave" the critical section, block the thread, and once the thread is released, "re-enter" the critical section before checking the loop body again.
3. Once a thread is done accessing the file and `current_sum` updated to reflect that, a number of threads could be blocked on `IDsum`. We release them all, as, depending on the values of `i, k`, and `current_sum`, none, one, many, or all of the threads currently blocked may be allowed to proceed. If not, those who cannot proceed will block themselves in the next iteration of the while loop.

Question 3. Deadlocks (10 marks)

1. A computer has six tape drives, with n processes competing for them. Each process may need two drives. For which values of *n* is the system deadlock free?

   Answer (4 marks):
   Up to 5 processes can use this system without a deadlock. There is always at least one process that can acquire two disk drives, allowing it to complete, releasing its disk drives, which will allow other processes to acquire the necessary set of disk drives, etc. For higher values of *n*, there is always a chance that six processes have one disk drive and are blocked, waiting for one of the other (blocked) processes to release one drive.

2. In a real computer system, neither the resources available nor the demands of processes for resources are consistent over long periods (months). Resources break or are replaced, new processes come and go, new resources are bought and added to the system. If deadlock is controlled by the banker's algorithm, which of the following changes can be made safely (without introducing the possibility of deadlock), and under what circumstances?
   - Increase Available (add new resources)
   - Decreases Available (remove resources permanently from system)
   - Increase Max for one process
   - Decrease Max for one process
   - Increase the number of processes
   - Decrease the number of processes

   Answer (6 marks):
   - Increase Available (add new resources): *no problem*
   - Decreases Available (remove resources permanently from system): *may lead to deadlock*
   - Increase Max for one process: *may lead to deadlock*
   - Decrease Max for one process: *no problem* (assuming that the process has not yet acquired all resources, i.e., HAVE = MAX)
   - Increase the number of processes: *no problem*, assuming processes start out with initially no resources
   - Decrease the number of processes: *no problem*, assuming the OS can correctly reclaim resources held by such processes

   Note: this question was a part of a question on the 3rd sample midterm I had posted (together with the answers). That was actually by accident ☹.

Question 4. Memory Management (10 marks)

A pure paging system (no segmentation) has a page size of 512 words, a virtual memory of 512 pages numbered 0 through 511, and a physical memory of 10 frames numbered 0 through 9. The current content of physical memory is as follows:

Physical Address	Content
0	……
…	……
1536	start of Page 34
2048	start of Page 9
…	…
3072	start of Page Table
3584	start of Page 65
…	……
4608	start of Page 10
…	……

a) Assuming that page tables contain frame numbers (rather than physical memory addresses), show the current content of the page table.

Answer (3 marks):

Page	Frame
.	
.	
.	
9	4
10	9
.	
.	
.	
34	3
.	
.	
.	
65	7
.	
.	
.	

b) Show the content of the page table after page 49 is loaded at location 0 and page 34 is replaced by page 12.

Answer (3 marks)

Page	Frame
.	
.	
.	
9	4
10	9
.	
.	
.	
12	3
.	
.	
.	
49	0
.	
.	
.	
65	7
.	
.	
.	

c) What physical address is referenced by the virtual address 4613?

Answer (4 marks)
Virtual location 4613 is virtual page 9, offset 5, which is at physical page frame 4, offset 5. The address is 2053.

Note: this whole question was also part of the 3rd sample midterm I had posted, together with the answer. That was by design ☺

Do not count on the same thing happening for the final exam. I can almost promise that I will not repeat a question from prior exams (actually Question 1 in this exam was already recycled from a previous final exam….)