PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Computer Science Department

St. George Campus

July 2018 EXAMINATIONS

CSC 369H
Instructor — Sina Meraji
Duration — 2 hours

Student Number: |_|_|_|_|_|_|_|_|_|_|

Last (Family) Name(s): _____

First (Given) Name(s): _____

---

*Do **not** turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below *carefully*.)

This Term Test examination consists of 4 questions on 11 pages (including this one). *When you receive the signal to start, please make sure that your copy of the examination is complete.*

If you need more space for one of your solutions, use the last pages of the exam and indicate clearly the part of your work that should be marked.

In your written answers, be as specific as possible and explain your reasoning. Clear, concise answers will be given higher marks than vague, wordy answers. **Marks will be deducted for incorrect statements in an answer.** Please make your handwriting legible!

Marking Guide

1: _____/16

2: _____/12

3: _____/12

4: _____/10

TOTAL: _____/50

Question 1. Short Questions [16 MARKS]

Part (a) [4 MARKS] what is context switching?

switch the CPU to another process, saving the state of the old process and loading the saved state for the new process

Part (b) [4 MARKS] what are the main 4 requirements of critical section problem?

1)Mutual Exclusion If one thread is in the CS, then no other is 2)Progress Ifno thread is in the CS, and some threads want to enter CS, it should be able to enter in definite time 3)Bounded waiting (no starvation) If some thread T is waiting on the CS, then there is a limit on the number of times other threads can enter CS before this thread is granted access 4)Performance The overhead of entering and exiting the CS is sm

This exam continues on the following page

Part (c) [4 MARKS] briefly explain what does $REQUEST\_SYSCALL\_INTERCEPT$ do in assignment1

you should know the answer :)

Part (d) [4 MARKS] Name a common page replacement algorithm that has relatively good performance and low cost? what is the main problem with LRU algorithm and what are the two alternatives for it?

Clock exact LRU is too hard to implement Alternatives
A) Randomly check the R bit
B) shift R bit to a counter, replace pages with large counter value

This exam continues on the following page

Question 2. Processes/Threads and Synchronization [12 MARKS]

At a child care center, regulations require that there is always one adult present for every three children. Below is an attempt at a solution to this problem (child threads execute the Child() function and adult threads execute the Adult() function). You should make no assumptions about the amount of time children spend at the daycare. Adult threads are considered to have entered the daycare when they execute their first V(), and to have left when they return from their last P(). Child threads enter when they return from their P(), and leave when they call V().

```
/* Shared global variables */
/* initial semaphore value is 0 */
sem_t *kids_allowed;
Child() {
   /* Enter the daycare */
   P(kids_allowed);
   play_or_nap();
   /* Leave the daycare */
   V(kids_allowed);
}
```

```
Adult() {
   /* Enter the daycare */
   V(kids_allowed);
   V(kids_allowed);
   V(kids_allowed);
   supervise_kids();
   /* Leave the daycare */
   P(kids_allowed);
   P(kids_allowed);
   P(kids_allowed);
}
```

Part (a) [4 MARKS] Show that the solution above enforces the requirement that there be at least 1 adult present for every three children

The semaphore value indicates the number of additional children that can be safely supervised in the daycare. Initially, it is 0, and no children can enter until an adult arrives. Once an adult arrives, it signals the semaphore 3 times, allowing up to 3 children to enter. A child may enter as soon as the first V() happens, since the adult is already present. Departing children cause no problems they may be replaced by another child, or may reduce the number of adults required. Departing adults are the only issue. For an adult to leave, it must first wait on the semaphore 3 times, effectively reducing the number of children that can be in the daycare. For these P()s to succeed, there must be 3 V()s, either due to children leaving (reducing the number in the daycare, and making it safe for the adult to leave) or due to another adult arriving (replacing the supervision of the adult that is leaving).

This exam continues on the following page

Part (b) [4 MARKS] Show how the solution above can prevent an adult from leaving, even though the regulations would permit it.

Suppose at some point there are 2 adults and 3 children in the daycare. The semaphore value will be 3, and 1 adult should be allowed to leave according to the supervision rules. The adults do not discuss their plans, and each decides to go home. Adult 1 calls P() twice, succeeding both times (semaphore value is now 1) and is interrupted. Adult 2 calls P() once, succeeding (semaphore value is now 0), then calls P() again and blocks. Adult 1 runs again, calls P() for the third time and blocks. Both adults are prevented from leaving the daycare, even though there are only 3 children to supervise.

This exam continues on the following page

Part (c) [4 MARKS] Solve the problem identified in (b). You may make your changes on the code boxes on the previous page if you prefer.

Introduce a new semaphore, dep_mutex with initial value 1. Adults leaving then execute:

```
P(dep_mutex);
P(kids_allowed);
P(kids_allowed);
P(kids_allowed);
V(dep_mutex);
```

Thus, departing is an atomic operation. Only 1 adult can attempt to leave at a time (the one that acquires dep_mutex), so the available child tokens can only be consumed by 1 of the adults. Note that there is an additional problem whereby new children can continue to arrive and prevent an adult from leaving this was not the intended problem to demonstrate in (b), but is also a possibility. In this case, children should also acquire the new mutex before trying to enter, so that once an adult begins leaving (gets the mutex) no new children can come in until they are done leaving.

This exam continues on the following page

Question 3. Scheduling [12 MARKS]

Consider the following workload:

Process	Burst Time
p1	13
p2	5
p3	23
p4	3
p5	31
p6	3
p7	14

calculate the average waiting time for each of teh following algorithms

Part (a) [2 MARKS] FCFS

$sum(0, 13, 18, 41, 44, 75, 78)/7 = 32.43$

Part (b) [4 MARKS] RR, q=8

$sum(47, 6, 56, 17, 61, 26, 60)/7 = 39$

This exam continues on the following page

Part (c) [2 MARKS] Shortest job First

$sum(0, 3, 6, 11, 24, 38, 61)/7 = 20.43$

Part (d) [3 MARKS] Which scheduling algorithm, as an operating systems designer, would you implement?

it is an opportunity for the student to give their views on scheduling algorithms. We discusses this in class and talk about pros and cons of different algorithms.

I would also give marks for saying that multi-level feedback queue scheduling would be a good choice as, by varying the parameters to this algorithm, it is possible to emulate all the other algorithms we considered. But the student should also say that even this is not ideal as vast amounts of testing and guesswork would still be needed. All implementing this algorithm does is give you the flexibility to try various algorithms. Many other answers are possible and the marks will be awarded on the basis of their argument and how they defend it

This exam continues on the following page

[1]

Question 4. Virtual Memory [10 MARKS]

Part (a) [4 MARKS] Suppose you had a computer that supported virtual memory and had 64-bit virtual addresses and 4KB pages. If a process actually uses 2^{26} pages of its virtual address space, how much space would be occupied by the page table for that process if a single-level page table was used? Assume each page table entry occupies 4 bytes.

There are $2^(64)/2^(12) = 2^(52)$ pages in the address space. Therefore, a single level page table would have $2^(52)$ entries, where each entry is 4 bytes. Therefore, the page table would require $4 * 2^(52)$ bytes . This is independent of how many pages are actually being used.

Part (b) [2 MARKS] Suppose, in the same machine as above, a two-level page table was used, such that the first level page table was four times the size of each second level page table. Do you have enough information to know exactly how much space is occupied by the two-level page table for that process? Explain

No. How the used pages are distributed across the address space of the process will determine how many second-level page tables are needed.

Part (c) [4 MARKS] If your answer to part (b) was yes, compute the space occupied by the two-level page table for that process. Otherwise, compute the minimum amount of space that could possibly be required for the two-level page table for that process and compute the maximum amount of space that could possibly be occupied by the two-level page table for that process. Be sure to show your work

[1]

This exam continues on the following page

First, we need to compute the sizes of the first and second level page tables. Since there are $2^(52)$ pages and the first-level page table is four times the size of each second level page table, the first level page table must have $2^(27)$ entries and each second level page table must have 2^25 entries (since their product must equal $2^(52)$). Further, since each page table entry occupies 4 bytes (2^2 bytes), the first level page table occupies $2^(27) * 2^2 = 2^(29)$ bytes and each second level table occupies $2^25 * 2^2 = 2^(27)$ bytes. Since $2^(26)$ second-level page table entries are needed, in total, to support the $2^(26)$ pages in use, there needs to be, at a minimum $2^(27)/2^26 = 2$ second-level page tables. These two second-level page tables, along with the first level page table, would occupy a total of $(2 * 2^(26)) + 2^(27) bytes$. At worst, each of the $2^(26)$ pages used by the process could be mapped by a different second-level page table (e.g. if the pages in use were spaced equally across the entire address space). In that case, $2^(26)$ second-level page tables would be required in addition to the first level page table, so the total space occupied by the page tables would be $(2^(26) * 2^(27)) + 2^(27) bytes$.

This exam continues on the following page