UNIVERSITY OF TORONTO MISSISSAUGA
DECEMBER 2014 FINAL EXAMINATION
CSC369H5F
Operating Systems Andrew Petersen
Duration - 3 hours
Aids: None

The University of Toronto Mississauga and you, as a student, share a commitment to academic integrity. You are reminded that you may be charged with an academic offence for possessing any unauthorized aids during the writing of an exam. Clear, sealable, plastic bags have been provided for all electronic devices with storage, including but not limited to: cell phones, tablets, laptops, calculators, and MP3 players. Please turn off all devices, seal them in the bag provided, and place the bag under your desk for the duration of the examination. You will not be able to touch the bag or its contents until the exam is over.

If, during an exam, any of these items are found on your person or in the area of your desk other than in the clear, sealable, plastic bag; you may be charged with an academic offence. A typical penalty for an academic offence may cause you to fail the course.

*Please note, you **CANNOT** petition to **re-write** an examination once the exam has begun.*

Last Name: _____

Given Name: _____

Student Number: _____

Signature: _____

This final examination consists of 5 questions on 18 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.

Please be sure that your handwriting is legible. Detailed answers will be given more credit than vague ones. Incorrect assertions will detract from otherwise correct answers.

Good luck!

Marking Guide

1: _____/12

2: _____/27

3: _____/10

4: _____/10

5: _____/16

TOTAL: _____/75

continued on page 2

24 marks **Question 1.** Short Answer

1 mark **Part (a)** From an operating system point of view, what is the problem with the etiquette rule, "Don't interrupt someone who is speaking"?

1 mark **Part (b)** From an operating system point of view, what is the problem with the driving rule, "If you arrive at a 4-way stop at the same time as a car arrives on the road to your right, the car to your right has the right-of-way"?

2 marks **Part (c)** **Caching** Define the term "caching" and explain what must be happening for it to be effective. Provide on example where caching is used in an operating systems context.

CONTINUED ON PAGE 3

1 mark **Part (d) CPU Scheduling** A CPU scheduling algorithm rewards a process whenever it returns from I/O by placing it at the front of the ready queue. Explain why this policy has been enacted.

3 marks **Part (e) System Calls** List the major steps that must be completed to execute and return from a system call. Begin with the execution of the system call stub on the user side and end with the return to the system call stub.

CONTINUED ON PAGE 4

2 marks **Part (f) Processes** In one or two lines, explain how a *thread* differs from a *process*. (Note: We are **not** using the OS/161 definition of "thread" in this question!)

2 marks **Part (g) Synchronization** Add appropriate synchronization to the following threads so that `resource--` is executed if and only if `resource` is greater than 0. Your solution should not enforce any other constraints. Be sure to declare and initialize any synchronization variables.

```
// Synchronization variables and initialization




// Thread X                          // Thread Y


resource--;                          resource++;
```

CONTINUED ON PAGE 5

2 marks **Part (h) Synchronization** Add appropriate synchronization to the following threads so that `statement xb` is executed after `statement ya` and `statement yb` is executed after `statement xa`. You may assume that this code will be called only once. Your solution should not enforce any other constraints. Be sure to declare and initialize any synchronization variables.

```
// Synchronization variables and initialization




// Thread X                          // Thread Y

statement xa                         statement ya



statement xb                         statement yb
```

2 marks **Part (i) Processes and Virtual Memory** Consider how fork is typically used today. On a uniprocessor system, is it better to schedule the child process to run immediately after a fork or to continue to run the parent process? Explain why. Be sure to take copy-on-write into account.

CONTINUED ON PAGE 6

1 mark **Part (j) Virtual Memory** In OS161, all entries in the TLB are invalidated on every context switch. Explain why. What other piece of information would be needed in the TLB to prevent the need to flush the entire TLB?

1 mark **Part (k) Virtual Memory** The "least *frequently* used" eviction heuristic evicts one the page currently in memory that has been used the least frequently when compared to other pages in memory. (If there is a tie, a random victim is selected from the set of pages that have been used least frequently.) Please explain why this heuristic cannot be implemented precisely.

2 marks **Part (l) File Systems and Security** Discuss the advantages and disadvantages of implementing permissions using (a) access control lists and (b) capabilities.

CONTINUED ON PAGE 7

2 marks **Part (m) File Systems** Describe how "." and the inode for "`file.txt`" are modified when the user executes "`rm ./file.txt`".

2 marks **Part (n) File Systems** The user has executed "`rm ./file.txt`". Under what conditions is it possible to recover "`file.txt`", and how would you do so?

CONTINUED ON PAGE 8

10 marks

Question 2. Concurrency

You are writing a program that uses threads to parallelize a task that must be completed many times. Please use mutexes, semaphores, and/or condition variables to implement the following requirements:

1. An initialization function will be called once, before any work is done.
2. To complete the task once, at least five threads must get to the marked "synchro point" in "subtask_A". Then one thread may start – and must fully complete – "subtask_B". Finally, all threads waiting at the synchro point in "subtask_A" (5 or more) may complete. This process will need to run multiple times.
3. Only one task may be in process at a time. That is, the thread executing "subtask_B" may not start until there are enough threads at the synchro point in "subtask_A" *and* until the previous threads completing "subtask_A" have exited the function.
4. Only one task may be in process at a time. That is, you should not start a "subtask_B" until all threads have exited the critical section in "subtask_A" (the code between "synchro point" and the end of the function).

```
// Synchronization variables




// Called once, before any threads are created
void initialize_synchronization() {




}
```

CONTINUED ON PAGE 9

```
void subtask_A() {

    // Synchro point
    printf("At the synchro point in A!\n");

}

void subtask_B() {

}
```

CONTINUED ON PAGE 10

Total Marks = 34

CONTINUED ON PAGE 11

*[Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*

CONTINUED ON PAGE 12

*[Use the space below for rough work. This page will **not** be marked, unless you clearly indicate the part of your work that you want us to mark.]*