UNIVERSITY OF TORONTO
Faculty of Arts and Science

DECEMBER 2017 EXAMINATIONS

CSC369H1F

Duration - 3 hours

Aids Allowed:
One double-sided 8.5 x 11 cheat sheet (printed or handwritten)

Student number: _____

UtorID: _____

Last name: _____ First name: _____

Lecture section: L0101 L0201 L2003 (circle only the one you are enrolled in)

---

*Do **NOT** turn this page until you have received the signal to start.*

(Please fill out the identification section above, **write your name, UtorID and student#, circle your lecture section**, and read the instructions below.)

This final exam consists of **8 questions on 22 pages** (including this one and pages that have intentionally been left blank). A mark of **at least 40 out of 100 on this exam is required to pass this course.** *When you receive the signal to start, please make sure that your copy is complete.*

Answer the questions **clearly and legibly**. Answers that include both correct and incorrect or irrelevant statements will not receive full marks. **Be careful what each question asks! Be specific rather than vague in your answers!**

We have provided next to each question, a rough estimate on how much time you should spend on it. These are just guidelines and add up to less than 3 hours. **Use your time wisely** and answer first the questions that you are confident about, then come back to those you might need more time to think about. **Do not panic! Relax and keep focused!**

If you use any space for rough work, indicate clearly what you want marked.

Q1:	_____/10
Q2:	_____/16
Q3:	_____/10
Q4:	_____/12
Q5:	_____/10
Q6:	_____/12
Q7:	_____/16
Q8:	_____/14
Total:	_____/100

Good luck!

[This page is left intentionally blank. 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.]

Q1. [10 marks - 1 mark each] True/False [5 minutes]

Indicate below, for each statement, whether it is (T)rue or (F)alse. Circle the correct answer.

T / F: Threads created within the same process share the same stack pointer and global variables.

T / F: Disabling interrupts is a generally-applicable method for ensuring mutual exclusion.

T / F: Acquiring a mutex or semaphore while inside a critical section can potentially lead to deadlock.

T / F: If no thread is blocked on a particular condition variable, then a signal on that condition variable will be recorded and let the first arriving thread go right through a pthread_cond_wait() operation on that condition variable without blocking.

T / F: Each process gets an equal scheduling quantum in the FCFS scheduling algorithm.

T / F: Priority inversion is the situation when a scheduler demotes a process to a lower priority queue, while promoting another process to a higher priority queue.

T / F: A child process inherits a copy of the parent's address space, with the pages marked as copy-on-write.

T / F: The working set of a process is used to model the dynamic locality of its memory usage.

T / F: A linked-based file system works well for random accesses, but poorly for sequential accesses.

T / F: To improve I/O performance on an SSD, defragmentation is recommended to improve locality of accesses.

Q2. [16 marks - 2 each] Reasoning questions – short answers [30 minutes]

1) [2 marks] Based on what we've learned throughout the course, how would you define an operating system? Be concise, but be specific.

2) [2 marks] Explain why it's necessary to validate user pointers for system calls.

3) [2 marks] Homer uses his own UNIX-based OS called DohOS, which keeps a system call table that stores pointers to the system call handlers. There are 100000 system calls in DohOS. Given the high number of system calls, Homer wants to save some space that the kernel takes up for the system call table. He figures that the routines to handle each system call can be selected using a switch/case statement, thus eliminating the need for keeping pointers in the system call table. Is this an acceptable solution - why or why not?

4) [2 marks] Describe one type of situation in which you would consider using processes over threads and why it would be beneficial to do so.

5) [2 marks] You are asked to consult for a company that performs big data analytics, whose application is slow despite using a high-end multicore machine with plenty of main memory. The application involves a set of components running in parallel in different processes, for security reasons. One of the application components is multithreaded to speed up computations by running them in parallel, and spinlocks are used to protect critical sections. The critical sections involve performing complex mathematical transformations on large chunks of data structures which are shared between all threads. Based on what you've learnt in this course, make a recommendation which could lead to improved application performance, and explain in detail why you chose this optimization.

6) [2 marks] For each of the two types of memory fragmentation discussed in this course, give a separate example of a scenario where that type of fragmentation would occur in a paged virtual memory system.

7) [2 marks] Consider that you have a hard disk for storing your files. Your data is stored in an ext2 file system in files of various sizes, and read ahead is enabled. Your daily activities involve reading or deleting existing files, as well as writing files, and after a few years, you notice that your system appears to be slower in terms of your daily activities. You considered upgrading your hardware, but you do not have any money. What action would you consider taking, to improve your existing system's performance? Explain the rationale for your action.

8) [2 marks] We know that a disk's rotational latency affects the speed of accessing data from disk. As a result, everyone should want a disk that operates at 54000 RPM instead of a disk which runs at only 5400RPM. Do you agree with this statement? Explain your rationale in detail.

Q3. [10 marks] Scheduling [15 minutes]

1) [3 marks] Consider the following set of processes which are not doing any I/O. Their arrival times and execution time units are included in the table below. The processes are scheduled on a single core, using a Round-Robin scheduling policy with a time quantum of 2 time units. If a process P finishes its time quantum at the same time as a new process Q arrives, then Q gets enqueued before P in the ready queue. Draw a diagram to show how the processes get scheduled (similarly to the one we discussed in class) and calculate the average wait time.

Process	Arrival Time	Execution time units
A	0	3
B	1	3
C	2	2
D	3	4
E	4	1
F	5	3

2) [3 marks] Explain why starvation is possible in the basic MLFQ scheduler and describe the strategy that the MLFQ scheduler can enforce in order to avoid starvation.

3) [4 marks] The disk scheduling queue contains this sequence of requests for block numbers:
12, 48, 57, 60, 65, 83, 155
The disk head is positioned on block #60 and the current direction is down (towards decreasing block numbers).
You can assume that no further disk block requests come in until all the remaining blocks in the queue are served. Which disk scheduling algorithm (FCFS, SSTF, SCAN, or LOOK) achieves the smallest seeking time for the scenario described above? Explain your answer by showing your calculations.

Q4. [12 marks] Synchronization [20 minutes + 10 minutes]

Use semaphores and mutexes to implement the *exact* semantics of condition variables in the empty structures and functions given below. Your solution for this will be marked for *correctness* and *efficiency*! The POSIX API for mutexes and semaphores is included on the last page of this exam paper. You may not declare pthread_cond_t variables (that would be cheating).

```
typedef struct _my_cv {









} mycv;

void cv_init(mycv *cv) {




}

void cv_wait(mycv *cv, pthread_mutex_lock *mutex) {









}

void cv_signal(mycv *cv) {









}
```

Q5. [10 marks] Address Translation with Page Tables [15 minutes]

In this question, we will perform address translations using the physical memory dump on page 25. Feel free to detach page 25-26 to avoid having to flip back and forth. Please DO NOT DETACH ANY OTHER SHEET.

The first translation exercise uses a linear page table, and the second one uses a two-level page table.

1) [1 mark] Warmup.

For a warmup, convert the following hex numbers into binary.

0x badfeed

0x facade

2) [4 marks] Linear page table translation.

Consider the linear page tables configuration as below:

- PTBR: 21 (i.e., the page table starts at the beginning of page frame #21)
- Page size: 16 bytes
- Virtual address space size: 1024 bytes
- Virtual address format:

[VPN5-0 | Offset3-0] => 6-bit VPN followed by 4-bit offset

- PTE size: 1 byte
- PTE format:

[PFN4-0 | V | S | D] => a Valid bit, an Onswap bit, and a Dirty bit, preceded by a 5-bit PFN.

Find the **data** stored at virtual address **0x0ff** by filling in the following table. Fill irrelevant fields with "N/A" (e.g., if it is a page fault).

Virtual Address (in hex):	**0x0ff**
Content of PTE (in hex):	
PFN of data (in decimal):	
Offset (in decimal):	
Content of data (1 byte in hex)	

[This page is left intentionally blank. 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.]

3) [5 marks] Two-level page tables.

Consider the two-level page tables configuration, as below:

- PDBR: 3 (i.e., the page directory starts at the beginning of page frame #3)
- Page size: 16 bytes
- Virtual address space size: 1024 bytes
- Virtual address format:

[PD_INDEX2-0 | PT_INDEX2-0 | Offset3-0] => 6-bit VPN, divided into a 3-bit page directory index, and 3-bit page table index, followed by 4-bit offset.

- PDE and PTE have the same format
- PDE/PTE size: 1 byte
- PDE/PTE format:

[PFN4-0 | V | S | D] => a 5 bit PFN, followed by a Valid bit, an Onswap bit, and a Dirty bit.

Find the **data** stored at virtual address **0x369** by filling in the following table. Fill irrelevant fields with "N/A" (e.g., if it is a page fault).

Virtual Address (in hex):	**0x369**
Content of PDE (in hex):	
PFN of 2nd level table (in decimal):	
Content of PTE (in hex):	
PFN of data (in decimal):	
Offset (in decimal):	
Content of data (1 byte in hex)	

[This page is left intentionally blank. 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.]

Q6. [12 marks] Memory management and TLBs [15 minutes]

1) [3 marks] When adding more memory to a system, the number of page faults either decreases or at most stays the same. Do you agree with this statement? Explain your rationale in detail.

2) [9 marks] Consider the following sequence of memory accesses (all are reads). You know that the TLB uses an LRU replacement policy and can hold a maximum of 2 page table entries. The memory replacement policy is FIFO and the physical memory can hold a maximum of 3 pages.
Fill out the outcomes for the sequence of memory accesses from the table below. You must select from the following possible outcomes for each cell:
a. TLB hit
b. TLB miss, memory hit
c. TLB hit, page fault
d. TLB miss, page fault

Virtual page accessed	Outcome
1	
2	
3	
4	
1	
2	
5	
1	
2	
3	
4	
5	

[This page is left intentionally blank. 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.]

Q7. [16 marks] File systems [25 minutes]

1) [6 marks] Assume that your course grades are stored in your file system in a regular file called "`mygrades.txt`" and that you are operating in the directory where this file is located.
Consider the following two ***independent*** scenarios:
S1) You run the following commands:

```
ln mygrades.txt ../backups/mygrades-copy.txt
cd ../backups
ln -s mygrades-copy.txt mygrades-copy2.txt
```

S2) You run the following commands:

```
ln -s mygrades.txt ../backups/mygrades-copy.txt
cd ../backups
ln mygrades-copy.txt mygrades-copy2.txt
```

Answer the following questions below, ***for each of the scenarios above***:
a) What happens if you remove the file `mygrades-copy.txt`? Explain in detail the implications.
b) What happens if you remove the file `mygrades.txt`? Explain in detail the implications.
Note: The two questions are *independent*. That is, for b) do not assume that `mygrades-copy.txt` has been removed.

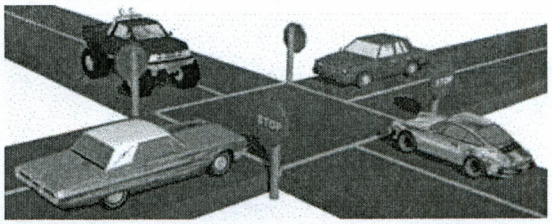
2) [3 marks] In ext2, what happens when we remove a file which happens to be stored as the first directory entry in the second data block of its parent directory.

3) [3 marks] How does fsck ensure that the data bitmaps match the data blocks which are in use by files and directories? Explain how this invariant is checked and what repair actions are taken.

4) [4 marks] You are consulting for a storage company that needs to ensure specific goals for their data storage system. Their **first goal** is redundancy in case of complete disk failures. At the same time, an **equally important goal** is good data bandwidth. The company currently stores their data in a RAID0 (striped) and you are asked to provide suggestions that would better suit the company's goals in terms of data storage. What would you recommend?

Q8. [14 marks] Synchronization bugs and deadlocks [20 minutes]

1) [6 marks] The diagram below illustrates a 4-way stop intersection.



Consider that cars can cross the intersection only straight through (no left turns or right turns), and must yield to cars coming from their right. Cars coming from opposite directions can go through the intersection at the same time.

The traffic monitor also has an "automated traffic cop", which only becomes active periodically. When the cop is inactive, the cars follow the original intersection rules described above. When the cop becomes active, all the cars that did not go through the intersection yet, must stop and follow the cop's instructions. The cop picks a random direction and signals all cars going in that direction to go through, ignoring the original intersection policies. Once all the signaled cars cross the intersection, the cop deactivates and comes back online again after a few minutes.

Explain whether a deadlock is possible or not, given:

a) the original intersection rules only

b) all the intersection rules (original rules, plus the automated traffic cop becoming active regularly)

Explain why the necessary deadlock conditions hold, *or* explain which one does not hold.

2) [3 marks] Banker's Algorithm. Consider that your system is currently running 4 resource-intensive processes: P0-P3, each having strict requirements in terms of 3 types of resources: A, B, and C. At this point, the processes have *already been granted* a number of resources of each type:

P0 (1, 3, 2) P1 (0, 2, 1) P2 (1, 4, 3) P3 (3, 1, 1)

For example, process P0 holds 1 unit of resource A, 3 units of resource B and 2 units of resource C.

The *number of resources available* (not allocated to any process) in the system is 2 of each type.

The *number of resources necessary* for each of the processes is:

P0 (3, 5, 3) P1 (7, 4, 2) P2 (4, 9, 6) P3 (6, 7, 8)

A process cannot complete without having all its resources met, but once it completes, all its allocated resources are released and can then be used by other processes.

a) Is our resource allocation system currently in a safe state? Explain your answer.

b) If in this state, P1 requests 1 unit of resource C, should this request be granted? Explain in detail.

3) [5 marks] A bank uses the following piece of code to perform a set of bank transfers from a given bank account ("src_acct") to a bunch of other bank accounts. The amounts to be transferred into each destination account are provided in the "amounts" argument. You may assume that the "transfers" function runs in a separate thread for every src_acct. You may assume the no self-transfers will occur (src_acct will not be included in the dst_list). Also, all accounts in dst_list are unique (no duplicates). Explain *in detail why* a deadlock could arise in this code *and propose a way* to change the code to prevent deadlocks from occurring (you do not have to rewrite the code, just explain what needs to change and how the changes would prevent deadlocks).

```
typedef struct {
  long acct_no;
  double balance;
  pthread_mutex_t lock;
} account;

void withdraw(account a, double amount){
  a.balance -= amount;
}

void deposit(account a, double amount){
  a.balance += amount;
}

void transfers(account src_acct, account *dst_list, double *amounts, int howmany){
  int i;
  pthread_mutex_lock(&(src_acct.lock));
  for(i = 0; i < howmany; i++) {
    pthread_mutex_lock(&(dst_list[i].lock));
  }
  for(i = 0; i < howmany; i++) {
    src_acct.withdraw(amount[i]);
    dst_list[i].deposit(amount[i]);
    printf("Transfer from %ld to %ld successful", src.acct_no, dst_list[i].acct_no);
  }
  for(i = 0; i < howmany; i++) {
    pthread_mutex_unlock(&(dst_list[i].lock));
  }
  pthread_mutex_unlock(&(src_acct.lock));
}
```

[This page is left intentionally blank. 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.]

[Feel free to detach this page, if you find it useful to avoid flipping to the question.]

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
page	0 :	00	ea	00	33	00	1f	7g	00	1e	ff	ae	21	19	77	19	0c
page	1 :	1b	1d	05	05	1d	0b	19	23	3d	00	10	1c	9f	09	00	1f
page	2 :	12	1b	0c	06	00	1e	04	13	0f	0b	12	02	1e	0f	00	0c
page	3 :	7f	32	3a	33	cd	7f	bc	77	3b	3d	ea	96	a4	11	42	7f
page	4 :	0b	04	10	04	05	1c	13	07	1b	13	1d	0e	1b	15	01	07
page	5 :	12	1e	15	7b	08	1b	35	0e	13	0f	1d	1d	1c	ed	12	0f
page	6 :	0b	0f	05	08	ae	00	7b	0c	0d	1e	00	00	00	00	00	00
page	7 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
page	8 :	11	10	1a	12	0f	10	18	0a	11	15	1e	15	1d	0c	12	17
page	9 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
page	10 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
page	11 :	09	10	14	cd	04	01	1f	18	17	0e	15	0c	05	0c	18	18
page	12 :	06	0b	16	19	1c	05	14	1d	01	14	1a	0a	07	12	0d	05
page	13 :	19	10	0b	0e	00	06	14	14	0f	1d	0e	09	1a	b9	12	15
page	14 :	12	18	14	0b	00	0d	1c	0a	07	04	b6	10	02	0c	14	c4
page	15 :	ab	11	c5	0e	0d	1b	08	15	4f	42	8a	09	09	1c	f9	2b
page	16 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
page	17 :	7f	7f	7f	7f	7f	7f	7f	7f	7f	7f	7f	7f	7f	7f	7f	7f
page	18 :	7f	7f	7f	7f	7f	7f	ab	7f	7f	7f	8e	7f	7f	7f	dd	7f
page	19 :	00	13	00	01	06	14	02	01	1e	0d	1b	06	be	0b	05	0a
page	20 :	1a	19	04	02	02	0c	1d	11	08	07	03	04	19	04	1a	42
page	21 :	0b	08	1b	0e	1c	15	1e	12	1e	05	0d	11	1e	11	1a	a4
page	22 :	17	13	1d	0a	12	02	11	19	06	08	15	07	08	1d	1e	b4
page	23 :	1d	0f	03	0f	0b	0d	7e	1e	11	13	14	0f	0f	ef	15	02
page	24 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
page	25 :	03	03	1c	03	1b	0e	0e	0a	1f	0b	11	0a	19	07	07	0e
page	26 :	09	0e	1d	18	08	5d	15	18	0d	0c	17	0d	07	0e	1d	04
page	27 :	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
page	28 :	0f	1d	0f	0a	02	11	07	0b	0b	17	07	1d	17	0e	1b	0b
page	29 :	17	08	1e	03	1b	01	07	10	12	0c	03	07	08	17	1c	12
page	30 :	7f	7f	7f	7f	7f	84	7f	7f	7f	7f	97	7f	bd	09	7f	f4
page	31 :	7f	7f	7f	7f	7f	7f	d0	7f	7f	7f	7f	7f	7f	7f	7f	7f

[Do not use this page for anything other than scratch work. THIS PAGE WILL NOT BE MARKED]

Reference API - the following functions are simplified (compared to POSIX API) for exam purposes:

```
pthread_mutex_lock(pthread_mutex_t *m);
pthread_mutex_unlock(pthread_mutex_t *m);
pthread_mutex_trylock(pthread_mutex_t *m);

pthread_cond_wait(pthread_cond_t *cond, pthread_mutex_t *m);
pthread_cond_signal(pthread_cond_t *cond);
pthread_cond_broadcast(pthread_cond_t *cond);

sem_init(sem_t *sem, unsigned int initial_value);
sem_wait(sem_t *sem);
sem_post(sem_t *sem);

//barrier will wait for num_threads when pthread_barrier_wait() is called
pthread_barrier_init(pthread_barrier_t *bar, unsigned int num_threads);
pthread_barrier_wait(pthread_barrier_t *bar);
```