Name: **NetID:**

CS 4410 Operating Systems Final, Fall 2019
Profs. Van Renesse and Schneider

KEEP CLOSED UNTIL START OF EXAM IS ANNOUNCED

- Turn off and stow away all electronic devices. You may not use them for any reason for the duration of the exam. Do not take them with you if you leave the room temporarily.
- This is a closed book and closed notes examination.
- **You have 110 minutes**. Do not leave the room unless you have received explicit permission. If you need to leave the room during the exam you may need an escort.
- You get 1 point if you fill in your name and NETID on this page. You get 1 point if you write your NETID in the top right corner on the remaining sheets. *DO IT FIRST.* You get 0 points for the whole exam if you do neither.
- Write your solutions within the provided boxes. Write clearly. There is an extra page if you don't have enough space in the provided boxes. Indicate clearly in the answer box if you have used this additional space.
- To receive partial credit you must show your work. State any assumptions you make.
- You can use the backs of these pages as scratch paper. *THEY WILL NOT BE SCANNED.*
- Turn in your work with the pages in order, ready to be scanned.

Question	Points
0: Name & NetID	2
1: Networking	10
2: Schedulers	15
3: RAID	12
4: Swimming Pool Monitor	12
5: Split Binary Semaphores	20
6: Page Replacement	12
7: File System	15
8: Security	7
9: Deadlock	8
Total	**110**

NetID:

Use this box if you do not have enough space in a provided answer box. Please indicate in the answer box that you have used this space, and in this space indicate what question(s) you are answering. DO NOT REMOVE THIS PAGE!

NetID:

1. [10 points] Consider two hosts connected through two routers as follows:

The links have the following characteristics (in both directions):

	One-way Latency	Bandwidth
Link L1	25 millisecs	1,000 Kbytes/sec
Link L2	125 millisecs	200 Kbytes/sec
Link L3	50 millisecs	1,000 Kbytes/sec

The hosts and routers do not add noticeable delay and can easily keep up with the attached links' bandwidths.

(a) What is the maximum achievable bandwidth from host H1 to host H2 in Kbytes/second?

200 Kbytes/sec

(b) What is the lowest achievable Round-Trip-Time (RTT) between hosts H1 and H2 in milliseconds?

400 millisecs

(c) When deploying a protocol like TCP, what size send window (in Kbytes) should each host use, assuming there is no other traffic?

200 x .4 = 80 Kbytes

Now suppose each link has a 0.1% loss ratio, i.e., one in a thousand packets get lost.

(d) What is the overall one-way loss ratio? (Giving either the formula or a close approximation is good enough.)

$1 - 0.999^3$ (approx .3%)

The end-to-end argument states that reliability guarantees implemented in the network generally cannot replace reliability guarantees implemented by the application and may make the former redundant. However, Ethernet and wifi both include mechanisms for detecting and retransmitting dropped packets.

(e) Was this a sensible decision by the Ethernet/wifi designers? Explain.

End-to-end, a packet drop can only be detected after 400 msecs. Any of the links individually can detect packet drops faster and thus provide faster recovery.

NetID:

2. [12 points] Scheduling

(a) Characterize the following schedulers with respect to starvation and interactive response. You may assume all jobs are of finite length. Ties in scheduling choices are resolved by uniform random choice.

	Makes starvation unlikely	Optimizes interactive response
Round-Robin	Yes	No
FIFO without pre-emption	Yes	No
Shortest Job First without pre-emption	No	No
Multi-Level Feedback Queue	No	Yes
Earliest Deadline First	Yes	No

(b) In pre-emptive scheduling, a scheduling decision must be made every time there is an interrupt. But performing a context switch on every interrupt is expensive, so pre-emptive schedulers like the Multi-Level Feedback Queue use "quanta" for a preferred interval between context switches. CPU-intensive jobs move down to lower-priority queues with longer quanta while I/O-intensive jobs move up to higher-priority queues with shorter quanta. In the following table, use checkmarks to indicate whether the currently running job should be moving down if possible (lower priority, longer quanta), up if possible (higher priority, shorter quanta), or kept at the same level as a result of the given interrupt.

	Move up	Stay the same	Move down
Clock interrupt in the middle of the job's quantum		✓	
Clock interrupt at the end of the job's quantum			✓
GUI interrupt for current job	✓		
GUI interrupt for another job		✓	
Page fault for current job		✓	

NetID:

3. [12 points] RAID-0 stripes blocks across multiple disks. Suppose you have four identical 1 GB (gigabyte) disks that each support reading and writing at 100 MB/sec. The mean-time-to-failure (MTTF) of each disk is 24 months.

(a) Answer the following questions about the storage array from the perspective of a user that treats the storage array as a disk.

What is the total usable storage capacity?	4 GB
What is the expected MTTF?	6 months
What is the maximum read throughput?	400 MB/sec
What is the maximum write throughput?	400 MB/sec

(b) To improve the MTTF, you investigate RAID-4 (block-level striping with dedicated parity disk) and RAID-5 (block-level striping with rotating parity). In either case, you would add a fifth disk, identical to the other four. Answer the following questions. Assume that, when the driver updates a block, it already has the previous version of the block and therefore it can compute the new parity block without having to read from any drive.

	RAID-4	RAID-5
What is the total usable storage capacity?	4GB	4GB
What is the maximum read throughput?	400 MB/sec	400 MB/sec
What is the maximum write throughput?	100 MB/sec	250 MB/sec

(c) Now suppose that you, instead, added a fifth disk that has the same capacity but is 4 times faster than the other four. You create a RAID-4 configuration with the fast disk as the parity disk. Answer the following questions:

What is the maximum read throughput?	400 MB/sec
What is the maximum write throughput?	400 MB/sec

NetID:

4. [12 points] The next page shows a solution to homework problem A3a, the pool problem. Recall, level is either 0 (middle school) or 1 (high school), so expression `!level` evaluates to "the other level" (`!0 = 1` and `!1 = 0`). Students invoke `pool_enter` to enter the pool and `pool_exit` to exit the pool. There can never be middle schoolers and high schoolers in the pool at the same time. Also, there can never be more than `NLANES` students in the pool. *Finally and importantly, a student that can enter the pool should not be blocked from entering.*

The implementation on the next page satisfies the following invariants:

```
0 ≤ count[0] ≤ NLANES
0 ≤ count[1] ≤ NLANES
count[0] = 0 ∨ count[1] = 0
```

In the following questions, "at line X" means "before executing the statement at line X". Answer the following questions with **yes** if the statement always holds, **no** if the statement can never hold, or with **maybe** if the statement can hold or not depending on the situation. You will get 0 points for the wrong answer, 1 point for no answer, and 2 points for the right answer.

at line 15, `count[0] + count[1]` may sometimes exceed `NLANES`	NO
at line 17, `count[level] = 0`	MAYBE
at line 17, `count[level] < NLANES`	YES
at line 22, `count[!level] = 0`	YES
in line 22, it would be incorrect to replace `cv_notifyAll` by `cv_notify`	NO
in line 25, it would be incorrect to replace `cv_notifyAll` by `cv_notify`	YES
line 24 can be deleted (making the `cv_notifyAll` statement on line 25 unconditional) without affecting correctness (only affecting efficiency)	YES

NetID:

```
 1  #define NLANES   7

 2  typedef struct {
 3    lock_t lock;
 4    cv_t cv[2];
 5    int count[2];        // #middle/high schoolers in the pool
 6  } pool_t;
```

```
 7  void p_init(pool_t* p){
 8    lock_init(&p->lock);
 9    cv_init(&p->cv[0], &p->lock);
10    cv_init(&p->cv[1], &p->lock);
11    p->count[0] = p->count[1] = 0;
12  }
```

```
13  void p_enter(pool_t* p, int level){
14    with(&p->lock) {
15      while (p->count[!level] > 0 || p->count[level] == NLANES)
16        cv_wait(&p->cv[level]);
17      p->count[level]++;
18    }
19  }
```

```
20  void p_exit(pool_t* p, int level){
21    with(&p->lock) {
22      cv_notifyAll(&p->cv[level]);
23      p->count[level]--;
24      if (p->count[level] == 0)
25        cv_notifyAll(&p->cv[!level]);
26    }
27  }
```

NetID:

5. [20 points] You are working with an operating system that only supports binary semaphores, but for your project you would like to use general (aka counting) semaphores. You decide to implement general semaphores using split binary semaphores. The provided binary semaphore has type `bin_sema_t` and supports methods `bin_sema_init`, `bin_sema_procure`, and `bin_sema_vacate`. Your code implements type `gen_sema_t` and supports methods `gen_sema_init`, `gen_sema_procure`, and `gen_sema_vacate`. Your code appears on the next page. Assume that `gen_sema_init` is invoked exactly once with a non-negative value before any calls to `gen_sema_procure`, and `gen_sema_vacate`. Assuming that `gen_sema_init` has been executed already, answer the following questions:

(a) Indicate whether each of the following predicates is an invariant (0 points for wrong answer, 1 point for leaving open, 2 points for correct answer)?

$0 \leq$ `bs_mutex + bs_wait` ≤ 1	YES
`bs_mutex = 0` $\wedge$ `bs_wait = 0` $\Longrightarrow$ at least one thread is executing within `gen_sema_procure` or `gen_sema_vacate`	YES
`bs_wait = 0` $\Longrightarrow$ at least one thread is blocked on procuring `bs_wait`	NO
the **boldfaced lines** in the code are part of the "critical section" and thus only one thread can be executing them at a time	YES

(b) Answer yes, no, or maybe ("at label X" means just before executing the statement at label X). 0 points for wrong answer, 1 point for left open, 2 points for right answer:

at label L0, `bs_mutex = 0` $\wedge$ `bs_wait = 0`	YES
at label L1, `bs_wait = 1`	NO
at label L1, `nwaiting = 0`	MAYBE
at label L2, `bs_mutex = 1` $\wedge$ `bs_wait = 0`	NO
at label L3, `value > 0`	YES
at label L4, `value > 0` $\vee$ `nwaiting > 0`	MAYBE
at label L4, `bs_mutex = 0` $\vee$ `bs_wait = 0`	YES
replacing the `if` statement in method `gen_sema_procure` by a `while` statement would change the semantics of the method	NO

NetID:

```
typedef {
    bin_sema_t bs_mutex;    // mutex semaphore
    bin_sema_t bs_wait;     // semaphore to block on
    int value;              // value of semaphore
    int nwaiting;           // number of blocked threads
} gen_sema_t;
```

```
// initialize general semaphore gs to given non-negative value
void gen_sema_init(gen_sema_t* gs, int value){
    bin_sema_init(&gs->bs_mutex, 1);
    bin_sema_init(&gs->bs_wait, 0);
    gs->value = value;
    gs->nwaiting = 0;
}
```

```
// split binary semaphore "V hat" method
void gen_sema_Vhat(gen_sema_t* gs) {
L0: if (gs->nwaiting > 0 && gs->value > 0)
        bin_sema_vacate(&gs->bs_wait);
    else
        bin_sema_vacate(&gs->bs_mutex);
}
```

```
void gen_sema_procure(gen_sema_t* gs) {
    bin_sema_procure(&gs->bs_mutex);

    // if general semaphore value is 0, block
    if (gs->value == 0) {
L1:     gs->nwaiting++;
        gen_sema_Vhat(sema);
        bin_sema_procure(&gs->bs_wait);
L2:     gs->nwaiting--;
    }

L3: gs->value = gs->value - 1;
    gen_sema_Vhat(sema);
}
```

```
void gen_sema_vacate(gen_sema_t* gs) {
    bin_sema_procure(&gs->bs_mutex);
L4: gs->value = gs->value + 1;
    gen_sema_Vhat(gs);
}
```

NetID:

6. [12 points] Given is the following reference string (sequence of pages):

1, 2, 3, **1, 4, 1, 2**

For this question you have only three frames in which to store the pages. For each of the FIFO, LRU, and OPT page replacement policies, figure out which of the last four references causes a page fault. The table below gives you enough room to also pencil in the frame contents after each reference, *but we will not use those for grading.*

Reference string ↓

	FIFO		*LRU*		OPT	
ref	frame contents	fault ?	frame contents	fault ?	frame contents	fault ?
1	**1**	**Y**	**1**	**Y**	**1**	**Y**
2	**1 2**	**Y**	**1 2**	**Y**	**1 2**	**Y**
3	**1 2 3**	**Y**	**1 2 3**	**Y**	**1 2 3**	**Y**
1	1 2 3	N	1 2 3	N	1 2 3	N
4	4 2 3	Y	1 4 3	Y	1 2 4	Y
1	4 1 3	Y	1 4 3	N	1 2 4	N
2	4 1 2	Y	1 4 2	Y	1 2 4	N

NetID:

7. [15 points] A proposed new Unix-like File System (UFS) consists of a superblock, a sequence of i-node blocks each with a sequence of i-nodes per block, and the remaining blocks used to store data. The block size is 4096 bytes (4KB) and block pointers are 32 bits. For a particular instantiation of UFS, the i-node structure is as follows:

	byte 0	byte 1	byte 2	byte 3
0	File mode	Link count	User ID	Group ID
1	File size in bytes			
2				
3	Time of last file modification			
4				
5	Direct block pointer 0			
6	Direct block pointer 1			
7	Direct block pointer 2			
8	Direct block pointer 3			
9	Direct block pointer 4			
10	Direct block pointer 5			
11	Direct block pointer 6			
12	Direct block pointer 7			
13	Indirect block pointer			
14	Double indirect block pointer			
15	Triple indirect block pointer			

(a) If you had a 1 Terabyte (2^{40} bytes) disk, how many blocks would you have? (2^x notation is ok.)

$$2^{40} \div 2^{12} = 2^{28}$$

(b) What is the maximum number of blocks that can be identified by 32-bit block pointers? (2^x notation is ok.)

$$2^{32}$$

(c) Given the i-node structure above, how many i-nodes fit in a block?
(Do *not* use 2^x notation.)

$$2^{12} \div 2^6 = 64$$

(d) The O.S. needs to look up the i-node number of "/bin". The data in the root directory consists of 10 blocks. Given that only the superblock is in the cache, how many disk blocks will the O.S. have to read at most? Explain your answer below by listing the sequence of disk blocks that will be read.

12

1. Read the block containing the root i-node
2. Read 8 direct data blocks
3. Read the single indirect block
4. Read 2 more data blocks

The DisksRus company has a client that wants to have a file system that spans 16 TB. However, the largest disk that DisksRus sells is only 1 TB and so the engineers hatch a plan to build a RAID-0 configuration with 16 disks. The engineers at DisksRus are divided about what to do about the software. There is a group that wants to change the UFS file structure and replace each block pointer with a pair consisting of a disk identifier and a block pointer local to the disk. There is another group that wants to leave UFS unchanged and claims it can build all support that it needs into the DisksRus disk driver.

(e) As a distinguished veteran engineer at DisksRus, which group will you support, and why?

I will support the second group, because we can use the first 4 bits of the block pointer to identify the disk and the remaining 28 bits to identify a block on the disk.

NetID:

8. [7 points] Security

(a) What are the three Au's in Butler Lampson's *Gold Standard for Security*?

1	Authorization
2	Authentication
3	Audit

(b) Rate Access Control Lists and Capabilities. Place "cheap" or "expensive" in each empty cell. 0 points for wrong answer. 1 points for correct answer. ½ point if left open.

	ACLs	Capabilities
Review rights for an object	CHEAP	EXPENSIVE
Review rights of a principal	EXPENSIVE	CHEAP

NetID:

9. [8 points] In a land far, far, away, there are a multitude of blind philosophers; all have taken an oath of silence. Five are randomly selected and seated at a lunch table, where a single fork is placed between each pair of adjacent seats. A seated philosopher, being blind, will not be able to ascertain where it is seating, which other philosophers are also seated at the table, nor an identity or other characteristics of the forks on left and on right.

The behavior of each philosopher can be described by the following program:

```
do forever
     - think
     - pick up either the left or right fork
     - pick up the other fork
     - eat
     - return both forks
end
```

(a) Is there a deterministic strategy for deciding which fork to pick up first that all philosophers can use to avoid deadlock, or would any such strategy lead to deadlock? Briefly explain.

Any strategy could lead to deadlock. There are only two deterministic strategies before eating, if the philosopher has no way to know its position, which other philosophers are at the table, and/or anything about the forks on left or right: (i) first pick up the left fork or (ii) first pick up the right fork. This strategy leads to a deadlock if execution is in lock-step and all philosophers happen to pick up their first fork at the same side.

(b) Assume all philosophers have adopted a strategy to first pick up the fork on the left, so deadlock is possible. Assume also that some forks are quantum-forks, although philosophers (being blind) cannot tell the difference between a normal fork and a quantum-fork. A *quantum-fork*, however, can be picked up and used concurrently both by the philosopher on its left and the philosopher on its right. If all forks were quantum-forks, then deadlock would no longer be possible. Quantum-forks are expensive, though, so what is the minimum number of forks that must be replaced by quantum-forks in order to avoid deadlock?

1