CS3214 Fall 2025 Final Exam Solutions

December 16, 2025

Contents

1 Networking (30 pts)

1.1 Know Your Internet (10 pts)

Determine whether the following statements related to networking are true or false.

(a) Internet is a portmanteau created from the words "interactive" and "network."

☐ true / ☑ false

The Internet is an interconnected network of networks, and that's where the word comes from.

(b) The Internet was so successful in accommodating new applications because it was designed to be a "dumb" network that focuses on packet delivery while being agnostic to their content.

☑ true / ☐ false

(c) Routing is the process of deciding to which outgoing link an incoming packet should be sent, whereas forwarding algorithms influence how this decision is made.

☐ true / ☑ false

It's the other way around.

(d) Congestion control is a cooperative mechanism primarily implemented in TCP and TCP-friendly transport layer protocols.

☑ true / ☐ false

(e) Access network providers often carry traffic to destinations that are outside their network without regard to geographic proximity between sender and recipient.

☑ true / ☐ false

(f) Propagation delay can be reduced with faster circuitry and/or better modulation or encoding techniques, e.g., by replacing 10Gbps fiber with 100GBps fiber connections.

☐ true / ☑ false

The propagation delay depends on the speed of light and the distance, and better technology will not reduce these.

(g) As network bandwidth grows, pipelined window protocols should use larger window sizes that increase the number of outstanding packets that have not been acknowledged, provided that there is sufficient space to buffer sent and received packets.

☑ true / ☐ false

(h) Transport layer protocols use port numbers to distinguish between different services at the application layer.

☑ true / ☐ false

(i) Google's statistic show that as of 2025, the vast majority of Google users connect to Google services via the IPv6 protocol.

☐ true / ☑ false

As shown in class and linked in the p4 handout, Google's stats still show under 50% IPv6 penetration, see https://www.google.com/intl/en/ipv6/statistics.html.

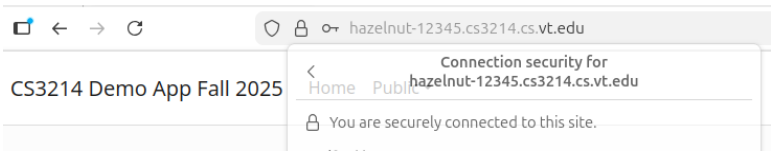
(j) In the economics of Internet transit, large Tier 1 providers such as Lumen Technologies pay smaller ISPs such as Shentel for the right to deliver traffic to their end customers.

☐ true / ☑ false

No, it is the other way around; smaller ISPs pay Tier 1 for transit.

1.2 Demoing p4 (12 pts)

When you demoed your project 4 in exercise 5, you asked the TA to visit a URL such as `https://hazelnut-12345.cs3214.cs.vt.edu`. The address bar in the TA's browser would have shown a lock symbol as can be seen in this screenshot:



If they had looked at their browser's network panel, they would have seen the following messages.

Message A:

```
GET / HTTP/2
Host: hazelnut-12345.cs3214.cs.vt.edu
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:145.0) Gecko/20100101 Firefox/145.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br, zstd
Connection: keep-alive
```

Message B:

```
HTTP/2 200
server: nginx/1.24.0 (Ubuntu)
date: Thu, 11 Dec 2025 15:39:09 GMT
content-type: text/html
content-encoding: gzip
X-Firefox-Spdy: h2
```

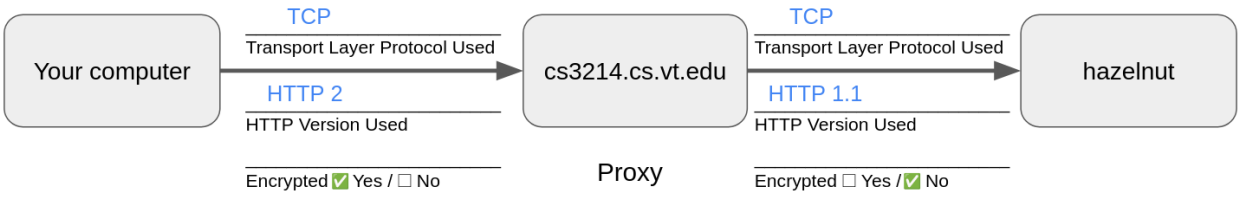
a) (2 pts) Which message contains a request, and which message is a response?

Message A is a request, and message B is a response

b) (1 pt) Which HTTP verb was used in the request?

The verb `GET` was used.

c) (6 pts) The following figure shows the setup used. Their computer did not connect directly to your p4 server, but rather it connected to a *proxy server* that functioned as a middle man.[1] Complete the figure by filling in the transport layer protocols used, the HTTP versions, and whether the connection was encrypted for each of the connections in the provided space:



[1] A similar setup was used if you demoed a deployment to the discovery cluster.

The properties of the connection to the proxy could be seen from the information provided in the problem; for the connection from the proxy to your server; hopefully, everyone remembered what they implemented a couple of weeks ago.

d) (1 pt) What is the name of the proxy server/software we used?

Based on the `server:` response header a software called `nginx`, see https://nginx.org/en/.

e) (2 pts) Given that a single proxy server was used for all students' demos, and given that students could run their demo on any rlogin machine with any port, describe how the proxy must process the `Host:` header of incoming requests.

The proxy used the value of the `Host:` header to find the name of the rlogin machine and the port at which it could connect to your server.

For those interested in the technical details, here is my setup for this:

```
server_name ~([a-z]+)\-([0-9]+)\.cs3214\.cs\.vt\.edu$;
set $fhost $1;
set $fport $2;
location / {
    proxy_pass http://$fhost-rlogin.cs.vt.edu:$fport;
    ....
}
```

1.3 A Visit to www.vt.edu (5 pts)

When Dr. Back recently visited the Virginia Tech website at www.vt.edu, his Firefox browser fetched 123 resources (JavaScript, CSS, images, etc.) from 17 domains, as shown in the following table:

#	Domain Name	Number of Resources
1	analytics.kaltura.com	1
2	applyto.graduateschool.vt.edu	1
3	assets.cms.vt.edu	1
4	bat.bing.com	1
5	cdnapisec.kaltura.com	7
6	cfvod.kaltura.com	4
7	connect.facebook.net	1
8	js.adsrvr.org	1
9	mx.technolutions.net	2
10	pixel.byspotify.com	1
11	pixels.spotify.com	3
12	script.crazyegg.com	1
13	static.ads-twitter.com	1
14	www.assets.cms.vt.edu	43
15	www.googletagmanager.com	11
16	www.redditstatic.com	1
17	www.vt.edu	43
		123

Assume that different domain names will resolve to different IP addresses.

a) (1 pts) Assuming that all domains support only HTTP/1.1, what is the least number of TCP connections the browser must open to load this page and its associated resources?

17 connections are at least needed.

b) (2 pts) The setting of `network.http.max-persistent-connections-per-server` in the Firefox browser used was set to the default value of 6 (note that this setting applies to HTTP/1.1 only). Assuming that all domains support only persistent HTTP/1.1, what is the largest possible number of TCP connections his browser would open to load this page?

The answer is the sum of the minimum of the number of objects per domain and 6, which is $1+1+1+1+6+4+1+1+2+1+3+1+1+6+6+1+6=43$.
(Note that this is the largest number; the browser will not open additional connections if it can reuse existing connections.

c) (1 pts) Suppose all domains support HTTP/2. How many TCP connections would be needed?

17 connection are needed - HTTP/2 opens only one connection per domain.

d) (1 pts) Suppose all domains support HTTP/3 over QUIC. How many TCP connections would be needed?

0 since QUIC doesn't use TCP.

1.4 JWT Key Management (3 pts)

In project 4, we provided a file `oct_key_256.json` for you to use in your demo. Its content is shown below

```
{
  "keys": [
    {
      "kty": "oct",
      "alg": "HS256",
      "k": "c3VwZXJzdXBlcnN1cGVyc2VjcmV0IGtleSBmb3IgZmFsbDIwMjU="
    }
  ]
}
```

If you ran its contents through the `base64` program, you would have seen:

```
$ echo 'c3VwZXJzdXBlcnN1cGVyc2VjcmV0IGtleSBmb3IgZmFsbDIwMjU=' | base64 -d
supersupersupersecret key for fall2025
```

If you changed this file before deploying your demo, tell us why. If you didn't change it, tell us why you should have (in other words, what are the security consequences of not doing so.) Be specific.

Knowledge of the key would allow an attacker to forge a JWT token and include it in a cookie, tricking the server into believing that the user is authenticated when in fact they didn't present their username/password.

2 Virtual Memory (18 pts)

2.1 Virtual vs Physical Memory Consumption (6 pts)

When examining a running process using the `top` command, the system administrator sees 300 MB under the virtual memory (VIRT) column but only 20 MB under the resident memory (RES) column.

Give two distinct scenarios why a process's virtual memory size could significantly exceed its resident memory size. For each reason, briefly explain what type of pages would contribute to VIRT but not RES.

Possible reasons included (needed only 2)

a) Reason 1: Demand paging: not all allocated pages have been accessed yet. For example, in a memory mapped file, pages that have not yet been accessed show up VIRT but not RES.

b) Reason 2: Paging out: some pages swapped to disk due to inactivity. For example, pages that have been evicted by the OS are still VIRT, but no longer RES.

c) Reason 3: (also OK) Copy-on-write: shared pages (e.g., in a forked child) that are not modified and don't have their own private copy yet. Technically, these shared pages will show up as "resident" in both processes, but since the total amount of memory used between both processes does not double count the shared memory, we will accept this answer.

2.2 Page Replacement (12 pts)

The following table shows a snapshot of a process's memory consumption at a given point in time. Assume that the OS was able to determine whether pages are actively used using an algorithm such as the clock algorithm.

Region	Permissions	VM Allocation	Resident Memory	Actively Used
Code Segment	`r-x`	20 MB	10 MB	10 MB
Heap	`rw-`	100 MB	50 MB	30 MB
Stack	`rw-`	8 MB	4 MB	4 MB
Memory-mapped file	`r--`	200 MB	100 MB	50 MB

If the system needs to evict 4MB worth of pages from this process due to memory pressure, which pages would be the best and worst choices to evict? Hint: consider expected page accesses and whether the pages are clean or might be dirty.

a) (4 pts) For each region, fill in numbers (1-4) to rank each as the best (1) to worst (4) choice for eviction:

__3__ Code Segment

__2__ Heap

__4__ Stack

__1__ Memory-mapped file

In order from best to worst

1: memory mapped pages that are resident and clean (`r--`) but not actively used (50 MB)

2: heap pages that are resident but may be dirty (`rw-`) and are not actively used (20 MB)

3: code pages that are resident and clean (`r-x`) but unfortunately actively used (10 MB)

4: stack pages that are resident and dirty (`rw-`) and actively used (4MB)

b) (4 pts) For the region you selected as the best choice (1) above, explain why:

As described above, the portion of the memory mapped file that is resident but not actively used can be freed without a need to write back and low likelihood of use.

c) (4 pts) For the region you selected as the worst choice (4) above, explain why:

As described above, stack memory is likely to be dirty and is being actively used, which means if evicted, it will need to be written to swap, but likely to be retrieved soon.

In summary, the OS should first evict inactive memory that doesn't need to be written back to disk, and last evict active memory that also must be written back to disk. The other two choices in between are inactive memory that must be written back to disk and active memory that doesn't; the OS should prefer the one-time write-back as it avoids future page faults provided the memory stays inactive.

3 Dynamic Memory Management (18 pts)

3.1 Exploring mimalloc (10 pts)

In their paper "Mimalloc: Free List Sharding in Action," Microsoft researchers Leijen, Zorn, and de Moura describe the design of their `mimalloc` allocator. For small requests that are less than or equal to $1,024$ bytes, this allocator is implemented as follows:

```
void* malloc_small( size_t n ) { // 0 < n <= 1024
  heap_t* heap = tlb;
  page_t* page = heap->pages_direct[(n+7)>>3];
  block_t* block = page->free;
  if (block==NULL) return malloc_generic(heap,n); // slow path
  page->free = block->next;
  page->used++;
  return block;
}
```

Answer the following questions:

a) (2 pts) Given that `malloc_small` is implemented as a thread-safe function designed for multithreaded environments, what kind of variable must `tlb` be?

It must have been a thread-local variable; otherwise the list manipulations in the function would constitute a clear violation. Note that making tlb merely an atomic variable will not work; the different threads' heap can't be shared.

b) (1 pt) Does this allocator use boundary tag headers?

☐ yes / ☑ no

For a boundary tag header method we would expect that the in-use bit in the header is flipped on an allocation, which is not the case here. There is also no update to any footer.

c) (1 pt) Does this allocator use singly-linked free lists or doubly-linked free lists?

☑ singly-linked lists / ☐ doubly-linked lists

Only `next` pointers are updated, no `prev` pointers, hence a singly-linked list.

d) (2 pts) How many size classes does this allocator use and what are their ranges?

The statement `(n+7)>>3` computes $\lceil \frac{n}{8} \rceil$. Therefore, the allocator rounds up to the next multiple of 8, and it uses $1024/8 = 128$ size classes for objects of sizes $8, 16, 24, \ldots, 1024$.

e) (1 pt) What is the maximum number of bytes lost to padding in each block after the payload?

7 bytes are lost in the worst case due to the rounding up to the next multiple of 8.

f) (1 pt) Does this allocator suffer from false fragmentation (that is, is it ever possible to have two adjacent free blocks that aren't combined into a larger free block)?

☑ yes / ☐ no

Yes, it suffers from false fragmentation - it uses a simple segregated storage approach. There is not enough information updated on an allocation to allow coalescing.

g) (1 pts) The initial free list from which `malloc` draws blocks is created in randomized order. What purpose does randomizing the order of free blocks in the free list serve?

Randomization protects against vulnerability exploits that require an attacker to predict where in the heap the next allocated objects goes.

h) (1 pt) The designers of `mimalloc` expended a great deal of effort towards optimizing their implementation, for instance, by reducing the number of branch instructions on the fast path. How many branch instructions occur in the assembly code of the given `malloc_small` function?

There is only one if-statement, hence only 1 branch instruction.

3.2 Realloc (4 pts)

Consider the standard `realloc()` function you implemented in project 3. Here is an example use:

```
void *newptr = realloc(oldptr, newsize);
```

When looking at this example from the perspective of the dynamic heap implementation, name two separate and independent conditions under which it may happen that `oldptr == newptr` is true.

a) (2 pts) Condition 1

If the new size of the block is smaller or equal than the old size, the block does not need to be reallocated.

b) (2 pts) Condition 2

The block to the right (in increasing address order) happens to be free and large enough such that the block that is reallocated can be extended into it, and the allocator's policies allow such coalescing.

We also accepted a scenario where the object is the last object in the heap and the heap is extended (since some of you implemented this in p3).

3.3 Memory Errors (4 pts)

Unsafe languages such as C are notorious for their potential for memory-related programming errors. Tools such as valgrind can identify these errors. Consider the following valgrind output, slightly shortened for clarity:

```
==38988== Memcheck, a memory error detector
==38988== Copyright (C) 2002-2024, and GNU GPL'd, by Julian Seward et al.
==38988== Using Valgrind-3.26.0 and LibVEX; rerun with -h for copyright info
==38988== Command: ./simple-errors
==38988==
==38988== Syscall param write(buf) points to uninitialised byte(s)
==38988==    at 0x496D157: write (write.c:26)
==38988==    (parts of the call chain are elided)
==38988==    by 0x401176: main (simple-errors.c:8)
==38988==
==38988== Invalid write of size 1
==38988==    at 0x40117F: main (simple-errors.c:9)
==38988==  Address 0x4a7a05e is 0 bytes after a block of size 30 alloc'd
==38988==    at 0x484682F: malloc (vg_replace_malloc.c:447)
```

```
==38988==    by 0x401157: main (simple-errors.c:7)
==38988==
==38988== Invalid write of size 1
==38988==    at 0x401192: main (simple-errors.c:11)
==38988==  Address 0x4a7a040 is 0 bytes inside a block of size 30 free'd
==38988==    at 0x4849B4C: free (vg_replace_malloc.c:990)
==38988==    by 0x40118D: main (simple-errors.c:10)
==38988==  Block was alloc'd at
==38988==    at 0x484682F: malloc (vg_replace_malloc.c:447)
==38988==    by 0x401157: main (simple-errors.c:7)
==38988==
==38988== (heap summary elided)
==38988== ERROR SUMMARY: 3 errors from 3 contexts (suppressed: 0 from 0)
```

Complete the C program that, when compiled with `-ggdb` and no optimizations, will produce those errors when run under valgrind:

```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  int
5  main()
6  {
7
8      fputc(*s, stderr);
9
10
11
12 }
```

One statement is already provided: four are missing (on lines 7, 9, 10, and 11).
One possible solution is given here:

```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  int
5  main()
6  {
7      char *s = malloc(30);
8      fputc(*s, stderr);
9      s[30] = '\0';
10     free (s);
11     *s = 'A';
12 }
```

Note that the byte values written on lines 9 and 11 are arbitrary. The type of `s` must be an 8-bit type, such as `char *` or `uint8_t *`.

4 Automatic Memory Management (18 pts)

In your answers to the following 3 questions, focus on the observable behavior of the program (what the user running the program could observe, for instance, if they started the program with `java -Xmx2g ScenarioA` on the command line).

4.1 Scenario A (6 pts)

Consider the following Java program:

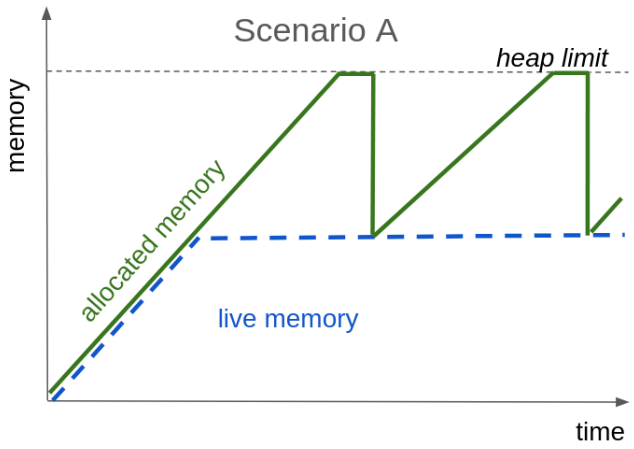
```
public class ScenarioA
{
  public static void main(String []av) {
    byte[][] l = new byte[1_000][];
    for (int i = 0; ; i++) {
      l[i % l.length] = new byte[1_000_000];
    }
  }
}
```

a) (1 pt) What will happen if a user runs program A while setting the maximum heap size to 2 GB?

The program will run forever/not finish/until the user terminates it.

b) (4 pts) Complete the following memory-time profile for this program under the conditions stated in part a). Use a dotted line for live memory and a dashed line for allocated memory. Assume that the garbage collection threshold is at or near the heap limit.

Scenario A's memory time profile might look like this:



c) (1 pt) What will happen if a user runs program A while setting the maximum heap size to 512 MB?

The program will run out of memory. (The live heap size is greater than $1000 \cdot 1,000,000$ bytes, which is clearly larger than 512MB.)

4.2 Scenario B (5 pts)

Consider the following Java program:

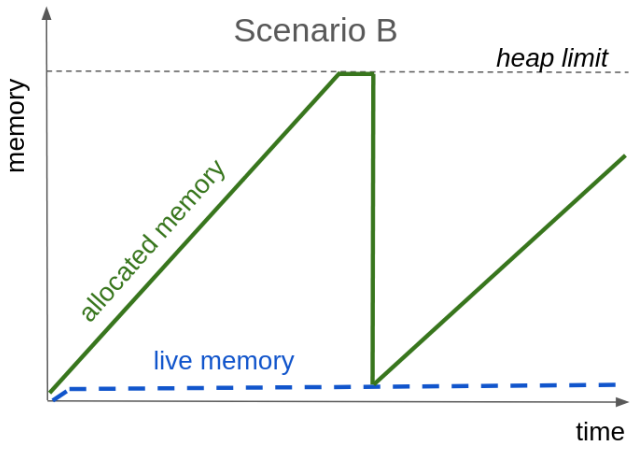
```
public class ScenarioB
{
  public static void main(String []av) {
    byte[][] l = new byte[1_000][];
    for (int i = 0; ; i++) {
      byte [] b = new byte[1_000_000];
    }
  }
}
```

a) (1 pt) What will happen if a user runs program B while setting the maximum heap size to 2 GB?

The program will run forever/not finish/until the user terminates it.

b) (4 pts) Complete the following memory-time profile for this program under the conditions stated in part a). Use a dotted line for live memory and a dashed line for allocated memory. Assume that the garbage collection threshold is at or near the heap limit. Use the same scale as in your answer to 4.1.

Scenario B's memory time profile might look like this. Note that this program keeps only 2 objects alive, so its live memory line should be substantially lower than in part A (near zero, in fact).



4.3 Scenario C (4 pts)

Consider the following Java program:

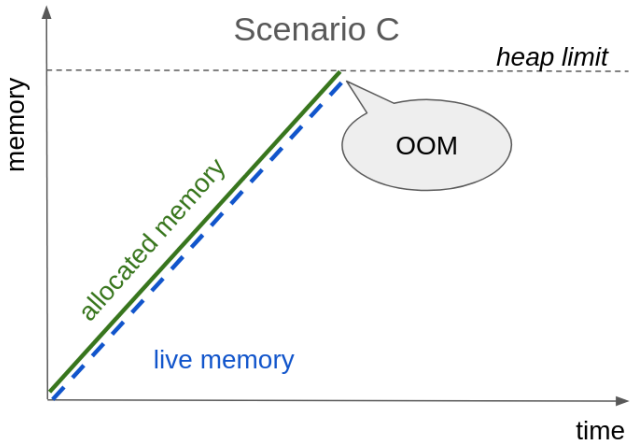
```
import java.util.*;
public class ScenarioC
{
  public static void main(String []av) {
    Set<byte[]> s = new HashSet<>();
    for (int i = 0; ; i++) {
      s.add(new byte[1_000_000]);
    }
  }
}
```

a) (1 pt) What will happen if a user runs program C while setting the maximum heap size to 2 GB?

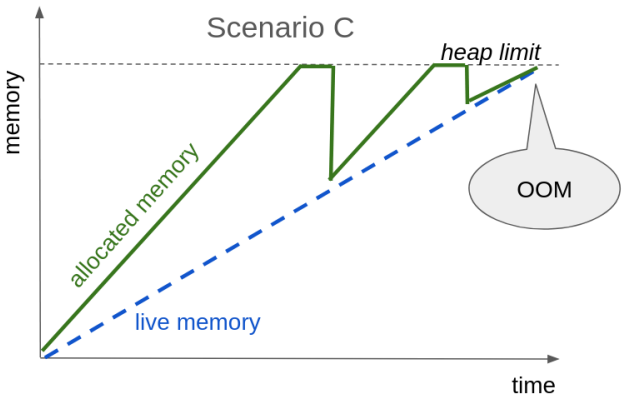
The program will run forever/not finish/until the user terminates it.

b) (3 pts) Complete the following memory-time profile for this program under the conditions stated in part a). Use a dotted line for live memory and a dashed line for allocated memory. Assume that the garbage collection threshold is at or near the heap limit.

Scenario C's memory time profile might look like this.



In reality, the live memory line might be lower than the allocated memory line, leading to some sawtooth pattern before running of memory, perhaps like so:



Both scenarioa were accepted as long as the live memory line was monotonically increasing to at or near the heap limit, and as long as the allocated memory was at or above the live memory line. Labeling the point where OOM occurs was not required, but it should be clear from the graph where it ends.

4.4 Identifying Leaks (3 pts)

Which of the preceding three examples contains a memory leak? Check all that apply.

Scenario A	Scenario B	Scenario C
☐	☐	☑

5 Virtualization and Cloud (6 pts)

1. In an Infrastructure as a Service (IaaS) model, the cloud provider is responsible for managing the operating system upon which user applications execute.

   ☐ true / ☑ false

   In IaaS, the customer manages the OS; provider manages hardware + virtualization.

2. Platform as a Service (PaaS) platforms typically provide developers with tools such as managed databases, build pipelines, and application runtimes.

   ☑ true / ☐ false

3. Software as a Service (SaaS) allows customers to interact with applications that are running and managed by the cloud provider.

   ☑ true / ☐ false

4. Elasticity refers to a system's ability to automatically scale resources up or down in response to changing demand.

   ☑ true / ☐ false

5. In a multitenant architecture, each tenant runs on completely separate physical hardware from all other tenants.

   ☐ true / ☑ false

   Multitenancy typically involves tenants sharing physical machines via virtualization or containers.

6. Processes are the preferred abstraction to prevent workloads from interfering with one another on the cloud because they utilize virtual memory.

   ☐ true / ☑ false

   Processes are in many ways insufficient to ensure workload isolation in cloud scenarios because they still share too many resources (namespaces, physical memory, etc.)

6 Essay Question: Rlogin or no Rlogin (10 pts)

Containers and container orchestration are widely used in industry today as a means of deploying applications in the cloud. Perhaps for that reason, our department's Techstaff has been arguing that they should no longer be required to support the rlogin cluster in the form it exists today: as a shared, non-virtualized, native computing cluster running Linux on 32 machines where Techstaff installs and maintains a consistent software environment. Instead, they propose to replace it with a (as of yet unspecified) "container-based" resource that allows (and requires!) students and faculty to develop and deploy their own container images for their CS3214 course work.
Based on your experience working with containers, discuss the merits and downsides of this idea.

Note: This question will be graded both for content/soundness of your technical arguments (6 pts) and for your ability to communicate effectively in writing (4 pts). Your answer should be well-written, organized, and clear.

Guidelines for graders.

To achieve 6 pts, we are looking for at least 3 separate arguments, which should include at least one downside and one merit. The arguments may include subjective perceptions and preferences as long as they are connected to technical justifications.

For the writing portion:

- (4 pts) The student writes in complete paragraphs and complete sentences and uses proper grammar and punctuation. The technical arguments can be clearly and easily understood and are presented in sufficient depth.
- (2 pts) The response lacks in clarity and/or organization and may not have been written using complete sentences, but the technical arguments can still be understood albeit with some difficulty.
- (1 pts) The student provided some answer, but due to its lack of clarity and organization it is difficult to understand the arguments being made.
- (0 pts) Did not answer.