Name	**UBID**	**Seat**

Question:	1	2	3	4	5	6	7	8	9	10	11	Total
Points:	10	5	5	5	5	5	5	20	25	25	25	100
Score:												

CSE 421/521 Final Exam

15 May 2017

Please fill out your name and UB ID number above. Also write your UB ID number at the bottom of each page of the exam in case the pages become separated.

This final exam consists of four types of questions:

1. **Ten multiple choice** questions worth one point each. These are drawn directly from the second-half lecture slides and intended to be (very) easy.
2. **Six short answer** questions worth five points each. You can answer as many as you want, but we will give you credit for your best four answers for a total of up to 20 points. You should be able to answer the short answer questions in four or five sentences. These are mostly (but not entirely) drawn from second-half material.
3. **One medium answer** question worth 20 points drawn from second-half material. Your answer to the medium answer should span a page or two.
4. **Three long answer** questions worth 25 points each integrating material from the entire semester. **Answer *only two*long answer questions.** If you answer more, we will only grade two. Your answer to the long answer question should span several pages.

Please answer each question as **clearly** and **succinctly** as possible—feel free to draw pictures or diagrams if they help. The point value assigned to each question is intended to suggest how to allocate your time. **No aids of any kind are permitted.**

I have neither given nor received help on this exam.

Sign and Date: ______________________________

Multiple Choice

1. (10 points) Answer all **ten** of the following questions. Each is worth **one** point.

   (a) Which university should GWA work at next?
   ◯ Illinois ◯ Northeastern ◯ Duke ◯ Stony Brook

   (b) Significant differences between file systems include everything *except*
   ◯ on-disk layout. ◯ reliably storing Vicky's data. ◯ data structures. ◯ crash recovery mechanisms.

   (c) What is a hint that a page might be good to swap out?
   ◯ It hasn't been used for a while ◯ It's currently loaded into a core's TLB ◯ Ali doesn't like it ◯ It's shared by multiple processes

   (d) Which of the following is *not* a useful approach to improving system performance?
   ◯ Carefully choosing an appropriate benchmark. ◯ Improving the parts of your code that you just know are slow. ◯ Analyzing data from experiments to identify bottlenecks. ◯ Developing a new simulator to improve reproducibility.

   (e) Which of the following is *not* a reason that virtualization became popular?
   ◯ Difficulty migrating software setups from one machine to another. ◯ Useful hardware virtualization features. ◯ Ability to reprovision hardware resources as needed. ◯ Lack of true application isolation provided by traditional operating systems.

   (f) A virtual address might point to all of the following *except*
   ◯ physical memory. ◯ a disk block. ◯ a port on a hardware device. ◯ a register on the CPU.

   (g) A correctly-implemented journaling file system will never lose data.
   ◯ True ◯ Only if Brijesh implemented it ◯ False

   (h) It is the *most* difficult to get repeatable performance results when
   ◯ asking Carl. ◯ running a system simulator. ◯ measuring a real system. ◯ using a system model.

   (i) Applications will run exactly the same in a virtual machine as they would on real hardware.
   ◯ True ◯ False

   (j) Which of the following makes it *easier* to virtualize the x86 architecture?
   ◯ hardware page tables ◯ instructions that are not classically virtualizable ◯ multiple privilege levels ◯ Guru

Short Answer

Choose **4 of the following 6** questions to answer. You may choose to answer additional questions, in which case you will receive credit for your best four answers.

2. (5 points) Name five different "names" that the operating system must virtualize to provide container virtualization.

3. (5 points) Draw a diagram of a traditional (or `ext4`-like) file system layout. Label the `inode` table (2 points) and show an example file, including `inode` (1 point), indirect block (1 point) and data blocks (1 point).

4. (5 points) Log-structured file systems rely on several assumptions to perform well. Identify one of those assumptions (2 points) and describe a realistic workload that would violate it (3 points).

5. (5 points) Describe the big idea behind Redundant Arrays of Independent Disks (RAID) (3 points). Identify another manifestation of that big idea in a different computer system (2 points).

6. (5 points) Explain the core cost-benefit tradeoff faced when swapping pages to disk. What is the cost (2 points)? What is the benefit, and how can it differ (2 points)? What is a clever way to reduce the swap-time cost to zero (1 point)?

7. (5 points) Define Amdahl's Law and describe how it guides the process of performance improvement.

Medium Answer

8. **(20 points) Virtualization Comparison**

   We discussed three types of virtualization in class: full hardware virtualization, paravirtualization, and OS or container virtualization. First, clearly describe each type of virtualization and give a brief overview of how it works (5 points each). You should explain what is virtualized, define the pieces of software that are involved, explain any constraints that this virtualization places on the virtualized environment, and identify any key challenges to this virtualization approach.

   Second, list three virtualization use cases that motivate each of the three approaches (2 points each, up to 5 points total). For each of your examples you should make a convincing case that the other virtualization approaches are impossible, ineffective, or perform poorly.

Long Answer

9. (25 points) Learning By Doing

A significant part of this class is completing the large programming assignments: `ASST2` and `ASST3`. Most of you will probably not work on these assignments again—although Carl has apparently implemented signals and completed ASST4. But hopefully you have learned some lessons from your struggles with OS/161 that you can apply to your future projects.

To answer this question, discuss **five** things that you learned by completing `ASST2` or `ASST3`. (If you did not complete or attempt either of these assignments, please answer the other two long answer questions.) In each case, describe what you learned and how you learned it (3 points) and how you will apply this lesson to your future projects (2 points). You can talk about tools, design, working with your partner, time management—whatever you want, as long as it is something that you learned this semester from the OS/161 assignments.

10. (25 points) Unikernels

Traditionally operating systems were designed to support multiple applications running together on the same machine. This was a reflection of an era when most machines—both desktops and servers—typically hosted many different users and applications.

Today, virtualization allows us to easily set up an entire machine dedicated to running a *single application*. Operating systems are beginning to evolve to support this use case. So-called **unikernels** are operating systems specifically tailored to supporting a single application: for example, a web or database server, or scientific application. The single application can run as normal using whatever operating system constructs it would normally use. However, the operating system below it has shed many of its traditional features as it only needs to support that single application. This question explores some of the implications and opportunities of the unikernel design.

First, being able to customize an operating system for a specific application places requirements on the design of the operating system itself. Identify and discuss these requirements (10 points). As a (big) hint, unikernels are usually built starting with so-called *library* operating systems.

Second, list three beneficial ways that the underlying operating system can be customized to support a single application (5 points each). For each, describe what is beneficial about that particular customization. To receive full credit, please include at least two different *kinds* of customizations.

11. (25 points) Asynchronous System Calls

Throughout the semester we have considered system calls as being *synchronous*, or blocking: when a process performs a system call, it is blocked until the call completes. However, modern operating systems also support *asynchronous*, or non-blocking, system calls. These allow a process to request the operating system perform some action on its behalf while not requiring the process wait for the action to take place.

First, considering the system calls we have discussed throughout the semester, describe several cases in which asynchronous system calls would be useful and how (3 points). Conversely, describe several synchronous system calls that lack a meaningful asynchronous analog (2 points).

Second, describe any changes to the operating system interface that might be necessary to support certain asynchronous system calls (3 points). Discuss any additional application programming challenges that non-blocking system calls may introduce (2 points).

Third, briefly explain how a process that can fork multiple threads can emulate asynchronous system calls without true non-blocking support from the operating system (3 points). Describe the overheads to this approach that might make native asynchronous system calls preferable (2 points).

Finally, describe the kernel changes necessary to support asynchronous system calls. Walk through the steps required to complete a non-blocking call, describing what happens at both the process (5 points) and kernel (5 points) level.