Sample 1

Faculty Development Project – Final Reports

[redacted]
Computer Science Department

February 20, 2018

1 Citizenship Project – Final Report

My Citizenship Project Proposal included two tasks:

1. Work with BYU CS faculty (on the 235 committee) to improve the CS235 course. We have begun meeting and are generating ideas.
2. Work with colleagues at Carnegie Mellon University, Tufts University, UMass-Lowell, and BYU on a MURI. The group has been assembled. A white paper is due in July and the full proposal is due in the Fall.

I'll report on my work and efforts on each of these tasks.

1.1 Report on Task 1 – CS235

I have been designated as the current chair of the CS235 curriculum committee. Other members of the committee are [redacted] and [redacted] each of whom has more overall experience than I do, though we are all relatively new to CS235. We have met and communicated quite a bit as a committee over the last half year or so. In so doing, we have looked at the course curriculum, the projects and activities we ask students to do in the class, etc. This alone has been a good experience for me, as I have been able to understand what others believe is important and how they think about CS education.

Team management has been just a little bit difficult for me. I believe that the difficulty arises in the fact that all people on the committee are quite knowledgeable and passionate about CS education. Nevertheless, they don't all have the same views. I think these differences are very good, though they do at times create a little bit of tension. I think that we have for the most part averted most of the tension, but we probably have not utilized each others' strengths perfectly. Ultimately, I think that everyone is doing well and enjoying their experiences teaching CS235. I hope that the students feel the same way.

One concern (and point of discussion in our committee) has been our ability to create a cohesive single offering of the course, while respecting each instructor's stylistic preferences. I think both objectives are really important. If we achieve one without the other we will be handicapping ourselves. I'm still thinking through how to best achieve this.

In short, this has been a good experience for me that I am still learning from. I believe that we have made good progress in understanding how student learn in this course, but have work to do.

1.2 Report on Task 2 – MURI with External Colleagues

I worked with [redacted] Seinfeld (CMU), [redacted] (CMU), [redacted] (UMass-Llowell), [redacted] (Tufts), and [redacted] Goodrich (BYU) to develop an ONR MURI proposal on the topic of "proficiency self-assessment" in autonomous systems. This proposal processed involved two steps. First, we submitted (in July) a white paper describing our initial ideas. Based on this white paper, we were invited to submit a full proposal on the topic. Overall, the people evaluating the proposals at ONR felt we had put together a very good project team, but they were less enthused about the proposal itself – they wanted us to adapt it somewhat. Thus, in the second step of this process, we put together a significantly modified, full proposal, which we submitted on or before November 1. We have not yet heard back on whether or not our proposed work will be funded.

For me, this was a good experience for a number of reasons, even if the proposal is not ultimately funded. First, since I'm still new to how funding agencies work in the US, I learned from this process. Second, it was nice to interact with researchers from other universities throughout this process. I think I came to know them and appreciate them

better than I had before. Finally, I think I learned a little bit about team management. In starting to the write the full proposal, the team got a little bit lost and confused about the objective of the research, with various team members "talking past" each other. In this case, one of the team members ([redacted] stepped forward and resolved the issue by presenting unifying ideas that allowed each person "place themselves" in the work. I thought this was skillfully done, and I learned from it.

My hope is that the project will be funded so that I can continue to work with these individuals in the coming years.

2 Scholarship Project – Final Report

In this report, I will address four things. First, I will report on my success in achieving each of the goals I made in the Scholarship Project Proposal I made for myself last June. Second, I will discuss my experience in applying the strategies I had created for reaching my scholarship goals. Third, I will revisit my research topics and themes related to my program of scholarship. Fourth, I will discuss what I have learned from this process and the plans I have to enhance my productivity with respect to scholarship.

2.1 Report on each goal

In June, I made the following three goals, for which I report my progress:

Goal 1 Publications: I had a rather ambitious goal (I thought) in which I hoped to submit (and hope for acceptance of) six specific papers for publication. Ultimately, I (and those I worked with) submitted and got accepted four out of these six publications, and I also have two other papers (which were not part of my initial goals submitted). The accepted publications include a paper in AAAI 2018 (with Mayada, my now-former PhD student), an article accepted and published in *Nature Communications* (which has received wide press and online attention), a paper with four of my former MS Students in HAI 2018 (this paper received a best student paper award), and a paper accepted to AAMAS 2018 (with Wen Shen, one of my former students). I had also hoped to submit a paper (with my current student Chace) to HRI 2018 and a workshop paper on the topic of "Composable AI," but those works have not progressed far enough to warrant submission yet. However, I did submit with one set of colleagues a paper to an HRI workshop (on narrative in longitudinal HRI), and a paper to EEE Trans. on Smart Grids with another set of colleagues (this paper is now entering its third round of reviews, and appears likely to be accepted). Thus, overall, I believe that this has been a very productive last 7 months for me with respect to publications. I am very grateful to those I have been able to work with and I am grateful that our efforts have led to successful results.

Goal 2 Grant Proposal Submissions: I had a goal to submit three different proposals: an ONR MURI with collaborators from various universities, an NSF proposal (likely to RI-Small), and a MEG (working eventually toward an NIH proposal). Working with collaborators, I achieved my goal with respect to submitting an ONR MURI (we have not yet heard if the proposal was successful or not). I did not end up submitting the NSF proposal I had intended, but I ended up working with collaborators in Mechanical Engineering on a separate NSF proposal (we have not yet heard if the proposal was successful or not). I also submitted a MEG proposal with a collaborator in the Department of Life Science, though it was not funded. Regardless, we continue to work toward the submission of an NIH proposal, though I think we are still a ways away. I also have plans to work with collaborators on submitting a proposal to a separate ONR project call (white papers are due in March). In short, I have been working on getting funding, but this is one area I feel I may need to emphasize a little more than I have been.

Goal 3 Tool development: I had a goal to work with my students to create a prototype "Composable AI" for document processing. We have made some progress, but not nearly as much as I would have hoped. This project is quite ambitious and difficult, and I perhaps bit off more than I could chew over the last 6-7 months. This goal was a casualty, though we are still working on it and hope to complete it (a prototype system) by the end of this semester.

2.2 Scholarship strategies

I listed three specific scholarship strategies in the scholarship proposal I created last June. I will list each one and then comment on them:

1. *Set aside substantial time to write:* This is sometimes difficult because of all the other duties that I have. I did succeed to some extent, as evidenced by the paper my colleagues and I have been able to write. I specifically set aside chunks of time in August-September (AAAI paper) and over the Christmas Holiday (Composable AI) to write, and I felt that both times were very important for the development of my research program. I also tried to consistently write during short intervals each week, though I must say that I have not always been successful in this regard due to other duties. I need to continue to set aside substantial time to write (including occasional bigger chunks of quiet time and more frequent "speed writing" sessions).

2. *Have fun working with students*: I particularly enjoyed working with Mayada on the AAAI paper. Jonathan also was involved in this process (running user studies), and that was fun. I have also really enjoyed working with the students on the Composable AI project, but that has not quite materialized yet, which puts a damper on the enjoyment a little bit. I need to and can do better on this emphasis, I believe.

3. *Work with the end in mind – refer back to the target deadlines and make course corrections to make sure I'm on track*: I have a list on the whiteboard in my office of the various deadlines and goals that I have. I think this has been helpful, and I have referred to it and revised it periodically.

Overall, I think these strategies have been good and helpful, and I plan to continue to use them.

2.3 Revisiting the themes and topics of my program of scholarship

Overall, I think my research themes and topics, as stated in my Faculty Development Plan are good. I do worry sometimes about getting too spread out, and I am making adjustments to them based on my time, level of interest, and anticipated ability to succeed in each project.

This is my current project emphasis going forward:

1. Developing narrative in human-AI interactions
2. Composable Artificial Intelligence
3. Interdisciplinary collaborations related to my work on human-AI interaction in repeated games.
4. Human-swarm interaction (perhaps phase out?)

2.4 Knowledge gained and future plans

Things I have learned or which have been reinforced to me:

1. Hard work and confidence pays off. As I work consistently, I can slowly accomplish what I set out to do.
2. There is a lot to do. I need go prioritize what is important. I do not have to do everything, but I need to do what I can do well.
3. I need to do a good job of working with others (students and colleagues both at BYU and outside of BYU). I'm more productive when I work closely and well with others. At the same time, I feel I need time to think and work independently to move some ideas along.

Plans for enhancing my future scholarly productivity:

1. Read the work of others. I need to do a better job of learning from the published works of others.
2. Help my students become more aware of past work.
3. I need to organize and stay on top of my student's efforts just a little bit better.

3 Course Development Project – Final Report

My course development project related to CS235 – Data Structures.

3.1 Course Background

Data structures covers the fundamental data structures and algorithms of computer science; basic algorithm analysis; recursion; sorting and searching; lists, stacks, queues, trees, hashing; and object-oriented data abstraction. This is the second CS course (after 142). It is taken by all CS majors (about 35% of the students), and it is also required for students in a variety of other majors.

I inherited the course as taught by [redacted] and have adapted it in small ways. By and large, however, the course is much like [redacted] left it to me.

3.2 Learning Outcomes

The learning outcomes for CS235 are as follows: By the end of the course, students will have greater capacity to solve problems by designing and implementing efficient computer programs. More specifically, students will be able to:

1. Use the fundamental data types of computing (lists, sets, maps, stacks, queues, priority queues).
2. Demonstrate understanding of the major techniques for implementing the fundamental data types (linked lists, binary search trees, hash tables, heaps) and implement several of them.
3. Properly select and use data structures from language-provided data structure libraries.
4. Apply basic algorithm analysis.
5. Demonstrate an understanding of how recursion works and write programs using recursion to solve problems.
6. Make informed decisions about which sorting and searching algorithms to use in specific circumstances.
7. Write programs with a size of about 500 lines of code.

I believe that these learning outcomes are all important in helping students understand the foundations of computer science. These learning outcomes speak to both the need for students to understand the basic operations of basic data structures, when they should use each of the data structures, and how they can implement (and, hence, deeply understand) these algorithms. These learning outcomes also speak to basic principles in analyzing algorithms, including computational complexity. The students will re-use recursion over and over throughout their careers, so it is important for them to begin to understand this technique. Finally, students in CS235 continue to develop maturity with respect to their ability to write computer programs that fulfill specific functions.

The course learning outcomes relate to the two BYU Computer Science Departments learning outcomes related to *computational practice* and *computational theory*. The course's learning outcomes related to selecting and using data structures, choosing sorting algorithms, and writing computer programs are all fundamental to computational practice. Meanwhile, the ability to perform basic algorithm analysis and understand the underpinnings of data structures relate to computational theory. Thus, each of the courses learning outcomes maps directly to the program's learning outcomes.

3.3 Course Activities

The class activities are:

1. Class discussions and lectures
2. Readings and lecture notes
3. Ten Homework
4. Eight programming labs
5. Lab help sessions
6. Exams (a midterm and a final)

These activities are designed to help students achieve the seven learning outcomes. The class discussions and lectures (along with readings and lecture notes) introduce the students to both computational practice and computational theory in computer science. These discussions and lectures prepare the students to do the homework assignments (which are designed to help the students know if they understand the topics, and hence are prepared for exams) and the programming labs.

Below, I map the course activities to each of the learning outcomes:

1. Use the fundamental data types of computing (lists, sets, maps, stacks, queues, priority queues) – Class discussions and lectures, programming labs 7 and 3
2. Demonstrate understanding of the major techniques for implementing the fundamental data types (linked lists, binary search trees, hash tables, heaps) and implement several of them – Class discussions and lectures, programming labs 2, 6, 8, and exams
3. Properly select and use data structures from language-provided data structure libraries. Class discussions and lectures, programming labs 2 and 3, and exams
4. Apply basic algorithm analysis. Homework, exams
5. Demonstrate an understanding of how recursion works and write programs using recursion to solve problems. Class discussions and lecture, homework, programming labs 5, 6, and 8, exams
6. Make informed decisions about which sorting and searching algorithms to use in specific circumstances. Class discussions and lectures, exams
7. Write programs with a size of about 500 lines of code. Class discussions and lectures, all programming labs

3.4 Assessments of Student Learning

The assessments used in the course are (1) exams, (2) programming labs, and (3) homework. Since the homeworks are just preliminary tests for the students, I do not use them as primary assessment instruments. Below, I specify the assessment tools for each learning outcome:

1. Use the fundamental data types of computing (lists, sets, maps, stacks, queues, priority queues) – This learning objective is primarily assessed by programming labs 3 and 7. If students can adequately complete these labs with 80% or higher, I consider that they have completed this objective. In general, if students are able to complete these labs then they have satisfied this objective.
2. Demonstrate understanding of the major techniques for implementing the fundamental data types (linked lists, binary search trees, hash tables, heaps) and implement several of them – Programming labs 2, 6, 8, and exams. Students should be able to complete these labs with a score of 80% or higher. The exams focus on this topic heavily, and thus students that get 75% or higher on the exams meet this objective.
3. Properly select and use data structures from language-provided data structure libraries. Programming labs 7 and 3, and exams. Students should be able to complete these labs with and 80% or higher score. The exams also focus on this topic, but we will use the programming labs for this assessment since I don't have scores on specific questions.
4. Apply basic algorithm analysis. Algorithm analysis is heavily tested on the exams. Students cannot score higher than 75% on the exams if they do not have a reasonable ability in this regard. Thus, the exams are the primary assessment instrument.
5. Demonstrate an understanding of how recursion works and write programs using recursion to solve problems. Programming labs 5, 6, and 8 all require an understanding of recursion. If students complete these labs at 80% or better, they have achieved this outcomes. Some exam questions speak to this, but I don't have data for individual questions, and therefore I will not use them as a formal assessment instrument.
6. Make informed decisions about which sorting and searching algorithms to use in specific circumstances. This learning outcome is heavily tested on exams. However, I don't have data that separates this out very well, since only some questions apply. In short, this is a weaknesses in my past offering of the course.

7. Write programs with a size of about 500 lines of code. If students complete the programming labs with 80% or better (overall), they have met this objective.

In general, I believe that these assessment instruments are effective. Two caveats: (1) We do not heavily policy cheating. Cheating is difficult on exams, but is possible on labs. Thus, the assessments are not cheating proof. (2) While we say that the students achieve a learning outcome if they meet the threshold, it is possible that they meet the outcome if they do not meet the thresholds as there are many confounding factors. Thus, in this sense, the assessments are somewhat conservative.

3.5 Student Achievements of Learning Outcomes

I address student achievements with respect to each learning outcome in turn:

1. Use the fundamental data types of computing (lists, sets, maps, stacks, queues, priority queues) – This learning objective is primarily assessed by programming labs 3 and 7. If students can adequately complete these labs with 80% or higher, I consider that they have completed this objective.

   Programming Labs 3 and 7: 81% of the students that completed the course completed programmings labs 3 and 7 with an average score of 80% or higher. Thus, this learning outcome was met for those students.

2. Demonstrate understanding of the major techniques for implementing the fundamental data types (linked lists, binary search trees, hash tables, heaps) and implement several of them – Programming labs 2, 6, 8, and exams. Students should be able to complete these labs with a score of 80% or higher. The exams focus on this topic heavily, and thus students that get 75% or higher on the exams meet this objective.

   Programming Lab 2: 151 out 170 students completed this lab at 80% or higher
   Programming Lab 6: 145 out 170 students completed this lab at 80% or higher
   Programming Lab 8: 138 out 170 students completed this lab at 80% or higher
   Just 7 of the 170 students did not complete any of these three labs at 80% or higher. Thus, with respect to this metric, the students achieved this learning outcome.

   Midterm exam: 123 out of 175 students had scores of 75% or higher. Final exam: 91 out of 167 students had scores of 75% or higher

   These results are not as high as I would like, and indicate holes in the students' learning.

3. Properly select and use data structures from language-provided data structure libraries. Programming labs 7 and 3, and exams. Students should be able to complete these labs with and 80% or higher score. The exams also focus on this topic, but we will use the programming labs for this assessment since I don't have scores on specific questions.

   Programming Labs 3 and 7: 81% of the students that completed the course completed programmings labs 3 and 7 with an average score of 80% or higher. Thus, this learning outcome was met for those students.

4. Apply basic algorithm analysis. Algorithm analysis is heavily tested on the exams. Students cannot score higher than 75% on the exams if they do not have a reasonable ability in this regard. Thus, the exams are the primary assessment instrument.

   Midterm exam: 123 out of 175 students had scores of 75% or higher. Final exam: 91 out of 167 students had scores of 75% or higher

   While most of the students met this objective, it is clear that some of them did not learn as much as I had hoped. One positive is that this is introductory material that is repeatedly covered in subsequent courses. Thus, this class has at least provided them with initial exposure to the topics that they can then perfect in subsequent semesters. Regardless, the course should be altered to better help all student achieve these learning outcomes to a high degree.

5. Demonstrate an understanding of how recursion works and write programs using recursion to solve problems. Programming labs 5, 6, and 8 all require an understanding of recursion. If students complete these labs at 80% or better, they have achieved this outcomes. Some exam questions speak to this, but I don't have data for individual questions, and therefore I will not use them as a formal assessment instrument.

   Programming Lab : 137 out 170 students completed this lab at 80% or higher
   Programming Lab 6: 145 out 170 students completed this lab at 80% or higher

Programming Lab 8: 138 out 170 students completed this lab at 80% or higher
Almost all of the students did not completed at least one of these labs at 80% or higher. Thus, with respect to this metric, the students achieved this learning outcome to a high degree.

6. Make informed decisions about which sorting and searching algorithms to use in specific circumstances. This learning outcome is heavily tested on exams. However, I don't have data that separates this out very well, since only some questions apply. In short, this is a weaknesses in my past offering of the course.

   Better assessment tools are needed for this learning outcome.

7. Write programs with a size of about 500 lines of code. If students complete the programming labs with 80% or better (overall), they have met this objective.

   128 out of 170 students completed all labs with an overall score of 80% or better. Thus, these students achieved this outcome.

In summary, these assessment instruments indicate that a large number of student achieved learning outcomes 1, 2, 3, 5, and 7. I do not have data to adequately assess learning outcome 6. Assessments of learning outcome 4 (basic algorithm analysis) indicates that many students did not become as proficient with this learning outcome as I hoped.

3.6 Steps Planned or Taken to Improve Teaching and Student Learning

Overall, I have mixed feelings about the results of the course. First, I believe that a large number of the students achieved each of the learning outcomes. However, a minority (but still substantial) number of students did not meet some of the learning outcomes at the level I would have liked. Thus, I think there is work to be done.

I'd like to make a few other observations about the course:

1. Overall, the student ratings for both sections I taught show that the students, by and large, felt they learned and had a positive experience in the course. Most student comments indicate that they learned from lectures and class time, and that they learned from the labs.

2. To me, there seems to have been a dichotomy in the students. Part of the students really caught on (either easily or through hard work), while the other another group just really struggled. I'd like to find out more about why this was the case. What separates the people in these two groups?

3. As I graded the final exam, I observed that some students didn't have the skills I thought they should have. This was particularly true of one question, in which students were asked to write a simple recursive function. Many of the answers weren't anywhere close. I felt we had gone over and over questions of this type in class and in assignments, so I was puzzled as to why some of the students were so far off.

4. I learned from survey questions sent to students that about a third of the students didn't feel they could write a simple C++ program if given enough time during the first week of class. Since we are assuming that they know C++ at least somewhat, this is a problem. I did track these students, and they ended up doing reasonably well in the class, though they were about a half letter grade lower than their peers up through the mid-term. We need to do more to help these students.

5. A good deal of the students 20-30% received D's or lower in the class. I'm not sure how to help these students. I spent a good deal of time reaching out to students in danger of failing. I think I can identify one or two cases in which these efforts paid off, but the efforts probably weren't successful in mass. I wonder if more effort should be given to helping the B and C students rather than the D and E students? Perhaps the B and C students are the ones that are struggling but continue trying?

Something things I'd like to do to improve the course:

1. In response to point 3 above, what can be done? I wonder if the students need more smaller assignments or activities with immediate feedback. I'd like to find a way for students to test and build their abilities with respect to simple concepts (without fear of getting graded).

2. I tried to spend more time in class implementing solutions this time than I did in previous semesters. In student evaluations, many students expressed appreciation for this. They seemed to particularly like when I made mistakes and had to work to find solutions – they could relate with this. I could probably do more of this.

3. I really try to encourage the students to participate in class. Overall, I do get pretty good participation, but classes are large, and many students don't or can't participate. I'd like to find ways to get more (and different people) to participate in class. I'd particularly like to get introverts and people who are really struggling to feel comfortable voicing their opinion (or indicating that they are lost).

4. Specific assessments tools are needed for learning outcome 6. I can resolve this by just tracking students' performance on specific exam questions.

Sample 2

Course Development Project – Final Report
Winter 2018

[redacted]

Course Background
In this course students learn about the elements of Computer Systems, including processes and process control, virtual memory, concurrency, network programming, and parallel programming. Most of the students taking this course are juniors and seniors. However, the programming concepts that most have learned to this point don't involve such low-level interaction with the operating system kernel, so the course presents a new paradigm and environment to which most are not very familiar. Many are used to developing with graphical tools and are less familiar with command-line use. Also, the asynchronous approach to many of the programs that are being designed make it non-intuitive in the more common procedural programming paradigm.

I have worked with other professors in the creation and development of this course, including one who teaches the same course in odd semesters and two who trade off teaching the prerequisite systems course. The course material is based largely off content developed by CMU professors, who wrote the textbook, though it has been adapted to BYU curriculum.

Learning Outcomes

- Linux Systems Programming
    - Students will demonstrate mastery of Linux system programming.
    - Students will be able to use Linux systems calls that are defined in C from within a C++ program to control processes and threads and to use shared memory synchronization, polling, file I/O, and sockets.
    - Students will be comfortable using the Linux command line and development environment to compile programs with Makefiles and for navigation, job control, I/O redirection, utility commands and manual pages.
    - Students will gain experience with advanced C topics (memory management, pointer manipulation, strings, etc.)
- Multiprocessing, Concurrency, and Exceptional Control Flow
    - Students will demonstrate mastery of multiprocessing, concurrency mechanisms, and exceptional flow control.
    - Students will write concurrent programs that provide efficiency gains from multiprocessing and demonstrate proficiency in threads, processes, mutual exclusion primitives, deadlock avoidance, signal handling and exceptional flow control.
    - Students will understand how the operating system implements processes, threads, concurrency mechanisms, including how the operating system provides scheduling.
    - Students will understand signal handling, exceptional flow control and virtualization.

- Virtual Memory
    - Students will demonstrate how virtual memory works and how it can affect program performance
- Internet Programming
    - Students will demonstrate a mastery of Internet programming, including using sockets, protocols, and non-blocking I/O.
    - Students use the BSD socket API to implement client-server programs using non-blocking I/O and polling to scalably multiplex large numbers of sockets.
    - Students use effective testing techniques for networked programs, including stress testing, load testing, and fuzz testing.
    - Students understand the role of TCP, IP, HTTP, DNS, and TLS as they relate to Internet services.

The learning outcomes embody concepts that are at the very heart of computer systems, and by learning these concepts, the students have a better understanding of not only programming but the systems on which programs run, how to most efficiently use those systems as both a user (i.e., of the command-line) and a developer.

Course Activities

The graded learning activities consist primarily of 1) homework assignments and 2) labs. The homework assignments are generally hands-on assignments, designed to guide a student through a given topic, either to program concepts that we have learned in class or to otherwise analyze and build on programs already written for the purpose of learning the concepts. Sometimes the homework assignments consist of problems from the textbook, designed to help them think about and solidify the concepts. The labs are larger projects in which the students implement concepts in a programming assignment, where they have a specification and are expected to produce a program that matches. The homework assignments are intended to help prepare the students for the labs and for learning assessments. The idea is to guide and teach students with the homework, lectures, and textbook reading, and then let them gain a deeper understanding of the concepts by applying them in the labs.

The labs and homework assignments are supplemented by in-class discussions and group activities. The former often occur at the beginning of my class. I post questions on the screen, and I ask the students to discuss the topic with their neighbors and answer them. The group activities were introduced later in the semester. I noticed students struggling with concepts and learning activities, and they could have benefitted from interactions with others. Ultimately, facilitating group discussion and conceptualization was an attempt to help the better performing students to help the students who were struggling.

Assessments of Student Learning

The course provides three major assessments of learning: two midterm exams and a final exam. The midterm exams cover approximately 75% of course material, and the final exam is comprehensive. The material on the exams is mostly based on concepts that they will have learned through the labs and homework assignments. The idea is that a student who has

completed the labs and homework assignments will at the very least have been exposed to and, in the best case, have acquired the knowledge not only to complete the learning activities but to truly learn the concepts in the process. The scores, therefore, will typically be a reflection of the knowledge acquired through the learning activities. And because the learning activities are focused on helping the students acquire the learning outcomes, it follows that the students that do well on the exams are internalizing the learning outcomes.

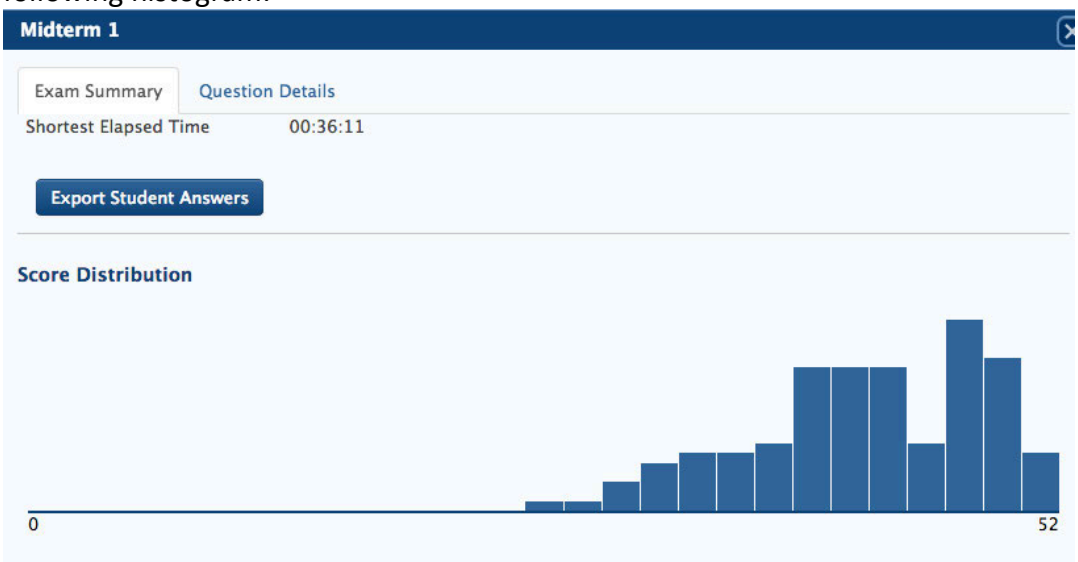
The format of the test was mostly True/False or Multiple Choice, with about 25% being short answer. The tricky parts about that are that 1) it's difficult—if not impossible—to give partial credit for incorrect answers; and relatedly 2) it's takes some effort to build large, multi-part questions whose overall value is appropriate for the material they cover.

Student Achievement of Learning Outcomes

The completion of the learning activities and the assessment of the students on the learning outcomes has demonstrated that students are achieving the course learning outcomes. At this point in the semester, the only midterm that has been administered covers only a subset of the learning outcomes of the course, but I show evidence below that students are demonstrating achievement of the learning outcomes.

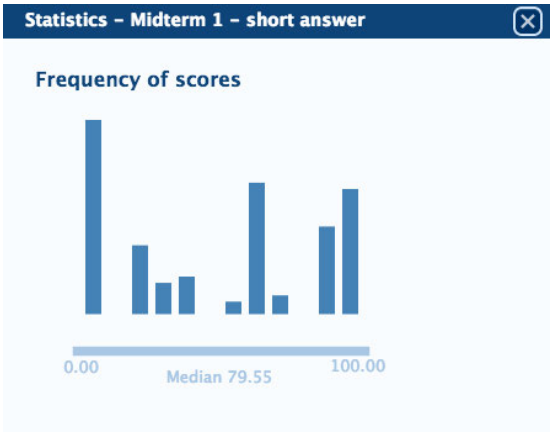
The first midterm covered some of the following learning outcomes: **Linux Systems Programming**; **Multiprocessing, Concurrency, and Exceptional Control Flow**; and **Virtual Memory**.

The distribution of combined scores on the first two of these learning outcomes is shown in the following histogram:



It appears from the scores that most students are getting a grasp of the concepts in the first and second learning outcomes.

Scores for the last topic, Virtual Memory, is shown in the following histogram:



Clearly there is work to be done to help the students with this learning outcome, as the assessment indicates performance that is less than expected. I also include in the appendix some students comments from a midcourse evaluation that indicates some success in this area.

Steps Planned or Taken to Improve Teaching and Student Learning

Many students have met with me in my office or have expressed through midcourse evaluations that the biggest challenges with this particular course are related to the concepts of the C programming language, Linux command-line usage, and memory management. My teaching colleagues, those teaching the same course in off semesters, and those teaching the prerequisite sister course, concur with that analysis.

In order to improve my teaching and the student learning in my classroom, one major change I intend to implement in future semesters is to have the learning activities at the beginning of the course include more introduction to foundational concepts that challenge students throughout the course. I specifically refer to the challenging concepts mentioned previously—C programming language concepts, Linux command-line usage, and memory management. Many students have met with me in my office or have expressed through midcourse evaluations that those are the biggest challenges with this particular course. The change to the activities at the beginning of the course will better help students with these challenges by allowing them to see the concepts hands-on and being a resource to refer back to for later learning activities.

I also implemented two changes mid-semester. The first, mentioned previously in this report, is the group activities. I observed that many students were struggling in part because they weren't interacting with other peers in the class to learn and apply the concepts. For the last three labs I am requiring group discussion and conceptualization, in an attempt to have the students that are better grasping the concepts to help the students that are struggling. But more generally, the goal has been to get students to work with others. I have already received anecdotal feedback that the change has been an effective one for the students.