Individual Instructor Report Fall 2022 Version A for C S 371P - OBJECT-ORIENTED PROGRAMMING (52930) (Glenn Downing)

Project Title: **Instructor Course Evaluations Fall 2022**

Courses Audience: **75**
Responses Received: **71**
Response Ratio: **94.7%**

Report Comments

Results were produced during the Fall 2022 implementation of new course evaluation questions and systems. Courses with final exams conducted on December 8th, 2022, may have overlapped with the evaluation window.

Guide to the Interpretation of Course Evaluations at UT Austin

The goal of course evaluation process at UT Austin is to drive teaching excellence and to support continuous improvement in teaching and learning experiences. The two sets of scales used for core evaluation questions and the associated weights are:

Strongly Agree (5)
Agree (4)
Neutral (3)
Disagree (2)
Strongly Disagree (1)

Excellent (5)
Very Good (4)
Satisfactory (3)
Unsatisfactory (2)
Very Unsatisfactory (1)

The Mean is calculated by adding all of the weights for a single question and dividing by the number of respondents. The course workload question is not averaged.

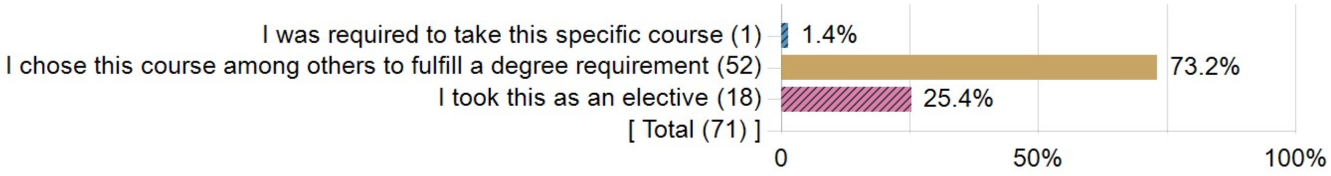
The number of students (e.g. Respondents) marking each option is reported for each of the items. These frequency distributions provide information about the level of student ratings and the spread and shape of the class distribution of responses. The distributions thus provide a picture of student perception of a course.

Course evaluations provide snapshots of student perspectives on their course-level learning experiences. Most experts on teaching evaluation advise that no one method gives the complete picture of an instructor's teaching effectiveness; multiple and diverse measures, on multiple occasions, are advised to give a full picture of the teaching effectiveness of a particular instructor. Moreover, other factors, such as size of class, level of the class, and content of the course, can cause small variations in the ratings. Therefore, student perspectives for a particular instructor or course should be interpreted as a snapshot, and as providing complete information on the teaching effectiveness of that instructor.

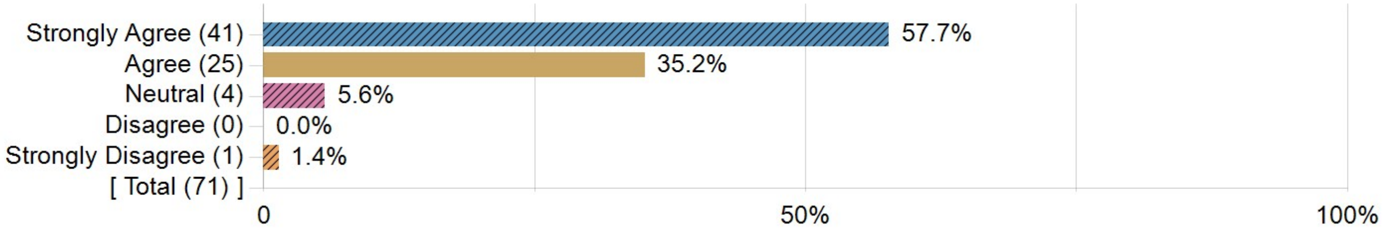
Creation Date: **Friday, January 6, 2023**

Course Questions

Why did you take this course?

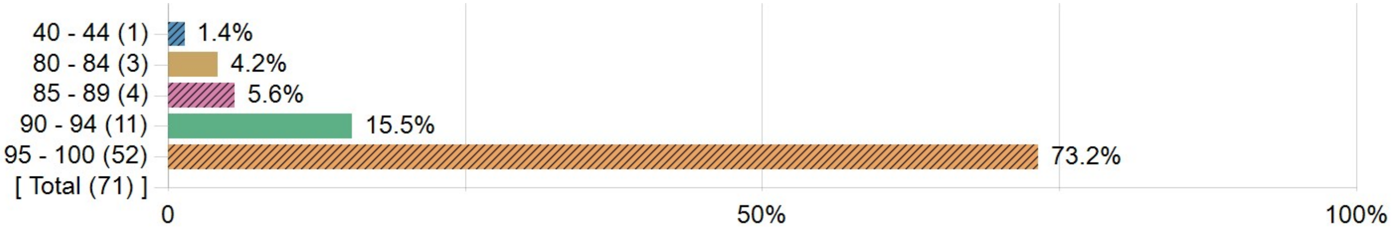


During this course, I gained a deeper understanding of the subject matter.



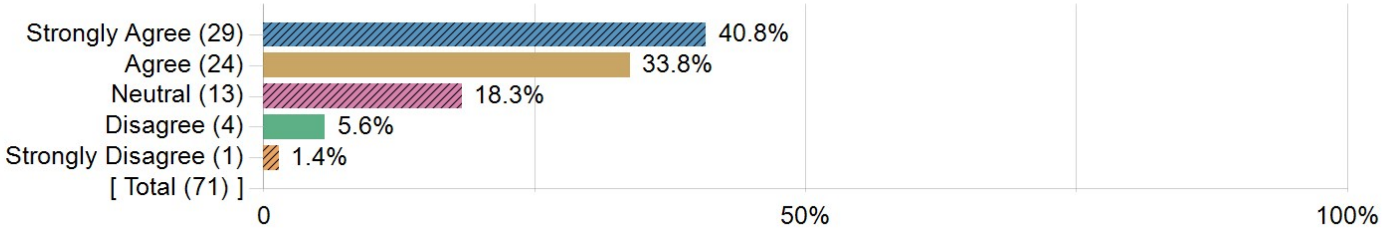
Statistics	Value
Mean	4.48

Overall, approximately what percentage of the course meetings did you attend or complete (online, in person, or asynchronously)?



Statistics	Value
Mean	94.59

The course challenged me to integrate skills and knowledge from my undergraduate coursework to undertake research or creative work.



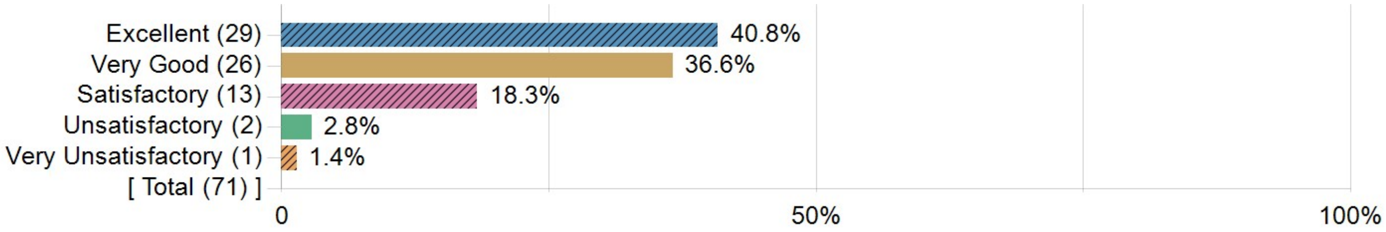
Statistics	Value
Mean	4.07

Instructor Questions

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree	Responded	Mean
The instructor clearly explained the course objectives and expectations.	66.2%	22.5%	9.9%	1.4%	0.0%	71	4.54
The instructor fostered an inclusive learning environment.	60.6%	28.2%	9.9%	1.4%	0.0%	71	4.48
The instructor effectively explained the concepts and subject matter in this course.	63.4%	25.4%	7.0%	1.4%	2.8%	71	4.45
The instructional techniques kept me engaged in learning.	47.9%	28.2%	18.3%	2.8%	2.8%	71	4.15
The instructor checked for student understanding of the concepts presented in the course.	66.2%	19.7%	8.5%	4.2%	1.4%	71	4.45

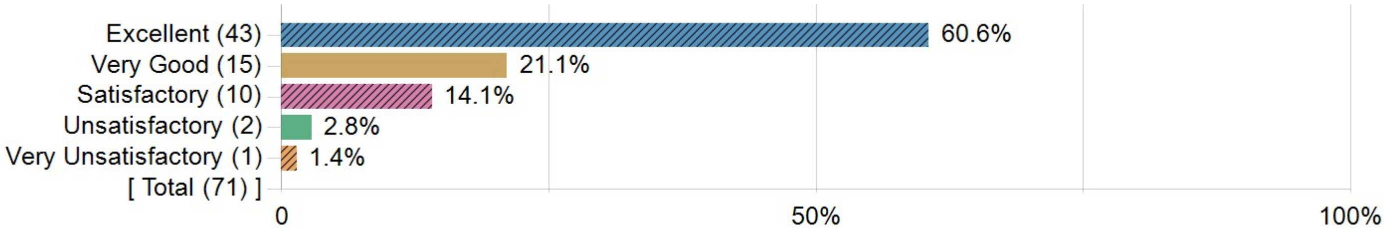
Overall Questions

Overall, this course was



Statistics	Value
Mean	4.13

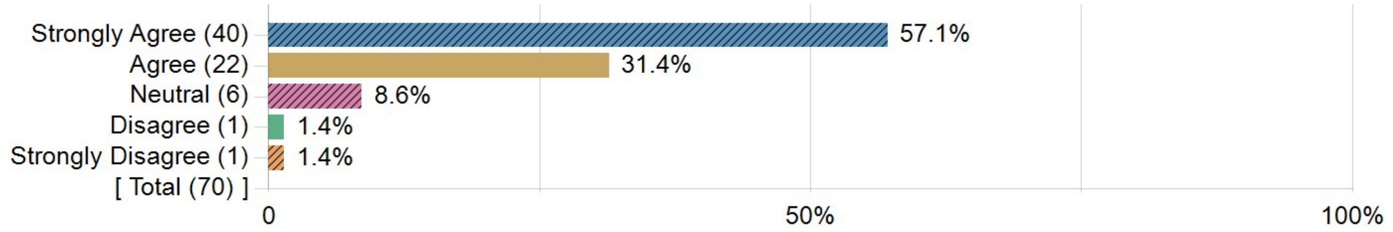
Overall, this instructor was



Statistics	Value
Mean	4.37

College, School, or Unit Questions

The instructor fostered a classroom environment in which all students could feel free to participate fully.



Statistics	Value
Mean	4.41

Comment Questions

Identify aspects of the course that were the most effective in helping your learning.

Comments
Dr. Downing was extremely knowledgeable on the subjects discussed in class and I am glad I took his course on Object–Oriented Programming. I learned things in this class that I did not even know that I did not know.
The course syllabus, while slow, gives a very thorough understanding of the material
Each lecture tended to have a concise focus with a good pace, and it was difficult to NOT come out of the lecture with the concepts cemented. I liked the cold–calling/conversations between the professor and the students, which leads to a more dynamic lecture. Exercises and Projects were definitely useful to reinforce the concepts. Finally, Ed Discussion and Discord helped to fill any gaps in understanding.
Professor Downing has a way with words, honestly. He articulates himself so well, and I think he does a really good job of understanding what exactly it is a newcomer to a subject matter needs to hear. He is truly a master educator, and one of the best professors at UT in my opinion.
The HackerRank Assessments during class were helpful in practicing the material we learned in class.
I found that cold calling was a really great way to stay engaged
This course filled in a lot of the gaps that I learned from other CS classes. It really helped me understand the intricacies of programming that were brushed over in other courses. Furthermore, this course also emphasized good coding practices within OOP such as CI, TDD, and the fundamental cornerstones of the paradigm. I enjoyed this class and it boosted my knowledge as a programmer. The concepts I learned in this course will be invaluable in industry.
Some of the effective aspects of the course definitely came from the methods Professor Downing used to help get the most out of people. Although I know a lot of people may find cold calling very daunting, I still personally feel that cold calling in the way that Professor Downing does (and with his very kind personality), often helps more than hurts. Downing is one of the most patient professors I have ever learned under. If someone was shaky on a certain topic, Downing was incredibly patient to make sure that person learns that specific section they got cold–called on (and more, sometimes).
He’s very nice and willing to spend time to help you.
The lectures were very well structured and presented. Professor Downing is an exceptional lecturer and is good at predicting and addressing questions that arise before they're even asked, or pulling them out of the student he's cold calling directly. The cold calling system solves the issue professors and other lecturers often have where they can't properly judge the room's understanding of certain topics; the person being questioned/ having the conversation instead acts like a rough measurement of the classes level of understanding, meaning that more or less time and detail can be applied to each topic. The projects were also integrated well into the course and taught programming techniques and methodologies that were legitimately useful and saved a lot of time debugging, in addition to promoting solid design. You could clearly see each topic discussed in the course directly lead into that section's project implementation and design, making the experience very cohesive.
Projects
– interactive class sessions – coding during class with examples – Perusall papers to emphasize important class concepts and teach new ideas –
I enjoyed the cold calling aspect.
The required reading and cold calling made me engage with the material more than in any other class I have taken at this school.
Projects, and exercises.
I really liked the exercises. They helped me get first hand experience and cement my knowledge. The projects were also very beneficial.
Lectures
Lectures and projects
The lectures were pretty helpful in learning some C++ semantics.
projects
The lectures were understandable and easy to follow.
Cold calling, excercises, daily quizzes, and PROJECTS were rlly good
The projects and in–class exercises
The cold calling method of teaching was most effective in both keeping me engaged and helping me understand the course material.

Comments
The projects
Professor Downing explained concepts in class very clearly. These concepts will most likely stick with me for years to come. I liked cold–calling as it made me more present in class.
I enjoyed the partner work and the interactive lectures.
I think the tri–weekly lectures were extremely helpful. In addition, I actually liked the cold calling, because I think it encouraged me to take a more active role in my own learning. In addition, Professor Downing was very god about being understanding when questions or objections were asked, and he also explained things at a reasonable pace.
The projects
I really liked how we didn't need to take notes and could just fully focus on the lecture. At first, I didn't think I would retain as much, but by the end of the semester, I was surprised by how much I learned.
The cold calling ensured I was paying attention during the course – I have a tendency to zone out.
I feel like the assignments were pretty helpful in my learning.
Professor Downing was a great lecturer and was able to explain things well.
Papers, projects, "cold–calling"
The in–depth explanation over each part of the content
The projects helped me in learning C++ and OOP principles.
Instructor was very thorough and explained everything many times. He reinforced learning. He also used various different assignments to make sure we learned the material.
Everything about the coding projects. They were approachable but complex. The two week time frame allowed us to take the time to think things through without being rushed and thinking "well, as long as it works and we can submit on time" which is the case with many of my classes. I really enjoyed working on the projects and I learned a lot from my partners
Projects
There was a bunch of independent learning to be done. It was extremely difficult at times, but I essentially learned an entire new language as well as much of its nuances all in a few months.
I appreciated the cold–call style lectures and the project–based learning for reinforcement.
The projects, while challenging, provide a deep level of understanding upon completion.
I really loved the socratic teaching style. It kept me extremely engaged and allowed to to view things from other peoples perspectives and hear other peoples thought process. I also felt like it created an open and friendly environment that encouraged us to ask questions.
Projects to a degree.
I think that having students involved in "conversation" during the lecture was cool as it forces everyone to stay engaged. It may also help students that are called upon that are confused on the current topic.
I liked the lectures. I really appreciated the thoroughness of the grading system design. It was easy to know exactly where I stood for the whole semester. I also thought the exercises were a lot of fun.
Lectures in class were very good at demonstrating concepts. The projects were also a good way to exercise the concepts that we learned in class.
The projects in particular were essential in reinforcing what we had been learning about in class.
Not allowing computers during class, cold calling, lecture recordings, daily quizzes
Certainly the projects. The lectures were also good.
I like that on projects we had to learn about how to compile the whole thing, linking files, and also working with the Makefile.
Cold calling, recording lectures for future references, working through concepts and problems together, questioning what we think code/methods/functions should do before revealing the answer
I think the aspects of the course that helped my learning is the exercises since they allow us to create well–designed programs without writing all the boilerplate.
Professor Downing did a great job in explaining the material and ensuring everyone in the class understood the material before moving on.
N/A
I really learned a TON about all the C++ language features. Before I had only done C and hastily learned C++, and now I can really understand all of the features of C++. I'm super proud!
I enjoyed the specifications grading style. It helped manage the stress that sometimes comes with cs courses.
Logical explanation of thought process

Comments
I think the lecture format was very engaging.

Identify the aspect of the course that you found most challenging, why you found it was challenging, and suggest one thing that could be done to help future students meet that challenge more effectively.

Comments
Some of the projects were much more challenging than the others, making it harder for me to plan the amount of time that I would have to dedicate to them.
I think that the no laptop rule both helped and hindered my ability to pay attention in class
I found having to make our own project structure, Makefiles, etc. was challenging at first, but very rewarding. I feel like I learned about a lot of new tools in the process.
I found the quizzes very challenging. The timing was very short, and the questions were often times unclear. That they were daily also made managing attendance difficult. I was sick for a few days, but made myself attend the class because I could not miss any quizzes. I also thought that the general grading scheme (the Specifications Grading) was challenging and unfair. The grading scheme does not take the average of different assignment scores, but the lowest as your final grade. It punishes students for their weaknesses and is discourages doing well in other sections. For example, a student that has gotten full credit on every assignment, but has a difficult time with the time–crunched quizzes will do poorly in the class because their work on the assignments no longer matter, since it's only the lowest score that counts.
The coursework in this class was very reasonable–– no complaints here!
I found it challenging when we did not have all of the expected output we were supposed to have in some of our projects. For example, in Darwin we were supposed to have the rows and cols be % 10, but this wasn't explained in the prompt. While not giving us this in the prompt emphasizes TDD, I believe in real life we will know the format we want our output to be before we start coding. It would be nice to know this ahead of time, as it's a trivial fix that could take hours to identify if not explicitly stated.
The most challenging part of the course was the quizzes that we took every class. They could be pretty nerve–wracking and for a class that had specification grading, requiring us to only miss eight (8) of the 41 quizzes was definitely pretty challenging. If you were out for a week due to sickness, you would immediately only have 5 quizzes to fall back on. In the end, I barely made the cut with only 7 misses, but that's undoubtedly due to some luck being on my side, as I guessed and miraculously got correct not only one but a couple of quizzes. Incredible.
I think the exercises were the most difficult thing.
The quizzes paired with Spec grading was really rough. I'd like to say I'm a relatively studious individual who takes notes every class and generally understands the material, but even I was struggling with the quiz questions. You essentially get "lives"; fail too many quizzes (get 2 or more out of the 3 questions wrong) and your grade drops to the next tier regardless of your other categories (unless you do even worse on those, but that's unlikely). Now, it may not seem difficult to get 2 out of 3 questions correct, but these are really tough questions that test your detailed understanding of the material, and the questions often build on each other. Very effective for learning, I agree, but they feel disproportionately impactful. I don't think lowering the quality of the questions is the right answer, or making them not graded (which would remove the incentive to do well on them). I'm not sure of an effective solution to this conundrum.
the quizzes, they are very fast–paced
The quizzes were the most challenging because of how nuanced and specific they were to the class concepts from the day before. It was difficult to find the one or two differences in a code snippet that would lead to a different output or quiz answer, which made it difficult to extensively prepare and do well consistently. One thing that I think could be done to help students in the future is to allow discussion in pairs or in groups and retake the quiz, as I have seen Professor Downing do in previous, virtual classes that he has taught (Spring 2022 SWE).
Nothing was insanely challenging.
I found it a bit weird at first that there was a disconnect between what was covered in lecture versus the projects. At times I felt that a more proper name for this course was "advanced C++"
Projects, not sure what to suggest.
I found all concepts of this course presented in a way that was easy to digest. I wish we had more help in how we can optimize performance for our code though.
Projects. Would have enjoyed coding assignments that have us use and implement the things we learn in class instead of projects. That way, quizzes are easier because we have a deeper understanding of how the material we're quizzed on works.
Exercises, time limit
The labs were challenging, but all for the wrong reasons. I don't really understand the point of hiding the output. As a software engineer, it's not like you don't get debugging help to find bugs in your code. It's not like industry professionals merely use unit/integration tests to figure out what's wrong. I don't understand why hackerrank debugging statements are hidden.

Comments
blogs, quizzes
The impact quizzes had on the grading scheme was probably the most challenging for me.
Quizzes were tough, cutoff was a bit strict for an A at least
being on time for the quiz, i think it should be at the end of class that way the content is fresh in my mind.
I had a very hard time finishing the in–class exercises in the allotted time frame. I wish we had the opportunity to either retry the exercise outside of class or have the deadline of the exercise be that day at midnight rather than during class.
The quizzes; reviewing lectures
It was challenging to understand some of the latter concepts introduced into the class given that they are more complex, but working with TAs helped. Going through notes and reviewing lectures talking about projects also help a lot.
Meeting an A on the class quizzes requirement felt stressful, I'd suggest there should be more leeway when it comes to the quizzes.
The grading system and slip days for attendance were dumb. It should have just been hybrid.
I thought the daily quizzes were extremely stressful. I am not a person who is able to parse through code and understand questions very fast, so I always felt pressured when those came up. However, I will admit that they were helpful in that I started watching past lectures before each class, thus allowing me to get a better grasp of the material throughout the course.
The quizzes sometimes didn't really relate to what was previously taught and a lot of them felt like "gotcha" quizzes. I believe if we were given a heads up on what notes/concepts from previous lectures to study before the quiz would be helpful.
I think the quizzes were the most challenging thing. Sometimes, it felt like there wasn't enough time to fully read, process, and answer all three questions.
It felt like there wasn't enough time to complete the exercises. It was always very hurried.
The in–class exercises were challenging for me. First, I think I didn't have enough time to pass the necessary tests. Whenever I'd ask for help, the TAs wouldn't be able to help me since they were too busy to get everyone's questions answered. Also, if the instructor could have fewer microaggressions toward his students, that would be great. Maybe the instructor means no harm with his questions and comments, but sometimes I thought they were a bit unnecessary considering they were not about the course material but about the student itself.
Throughout the course, I didn't know how many quiz grades I had that were a two or above and had to count them to know. A grade that counts the amount of quiz grades at a two or above would be helpful to know what my grade is.
I found understanding and true internalizing some of the papers to be somewhat challenging. I think a brief review of the topic covered at the start of the week might be nice, perhaps by integrating the papers into the daily quizzes.
The number of steps required for project completion. Building out a checkpoint/timeline tracker of when to have specific steps of the project completed.
Exercises, just because of the time limit. Maybe increase the time allowed for them.
The class was hard at the beginning because not much guidance was given. The assignments are slightly scattered and hard to start, but you get the hang of it.
The quizzes were challenging. I often felt like there wasn't enough time, and since we can't collaborate with partners anymore like this class used to do, it was easy to miss too many and everyone I talked to was most stressed about missing too many quizzes.
Quizzes, there were a lot of curve balls and trick questions that were challenging to parse.
The projects were an absolute doozy at times. There is just so much required in them that they took so many hours to complete. Even still, I would often forget 1 little thing and find myself right back at it for more hours. Little would be explained and we were expected to go figure it out.
The most challenging aspect of the course was probably the projects. Even though they are structured in such a way that we always know what to do, the design required to properly integrate takes some getting used to.
The quizzes often felt like trick questions and I believe it would be more helpful for them to be a summary of the last lecture than detailed questions about c++.
The daily quizzes were definitely the most challenging part, and the time crunch on the exercises. By the end I figured out (usually) the best way to prepare, but it took me a long time to find something that worked for me and helped me succeed so my quiz grades took a lot of hits earlier in the semester. I also felt like a few of the quizzes content wasn't taught or emphasized in lecture, they were maybe stated in one sentence and that made it challenging because sometimes what was said wasn't even reflected in the notes so it was hard to remember. Also I felt like the exercises were a little too rushed with their time limit sometimes. I also wish I would have been able to take notes on my ipad during lecture– so that way I could write down the things that weren't being typed on our shared class notes.
The actual learning. Everything could have been learned on an online course. The course did not help further my understanding, in fact it felt like the class was a constant play of beating around the bush instead of getting straight to the point of the concept. I understand this is a method of teaching to help us better retain information and understand why, however, if that was the case, I believe a quizlet with all this information could have been more effective than this course. The course was not as enriching as

Comments
reviews and other students have said it was to be. In fact, this class should be labeled as intro to C++ rather than object oriented programming. It would be a good lower division course as a supplement to data structures but not for students who have undergone the trials of computer architecture and operating systems. I feel as though I went back in time and was being taught these topics as if I was in middle school with the pacing of things.
I find that completing exercises was challenging at times due to the time constraints. I think some things that can help include being up–to–speed on the coding syntax needed for (usually whatever is relevant to the topic) and knowing when to ask for help.
I think you did really well with the class itself, and I never struggled with it. One thing I think you should change is this: at the beginning of the year, you called on a student and asked where they were from. They clearly were uncomfortable with answering the question, but you kept pressing it until they had to deliberately refuse to answer. We learned in the TA training that this can be interpreted as a micro–aggression and make the student feel targeted. I think you should be careful to avoid doing this in the future.
I was not a big fan of the spec grading in this class – it felt unfair to have only 8 drops out of 41 quizzes, especially when that includes being sick/being absent/etc. The quizzes are so fast, so I feel as if they're not always a great representation of someone's knowledge of the material. It doesn't feel fair to have done well on all of the projects but get a B in the class just because you did poorly on a few quizzes.
The topics covered in the papers could stand to be discussed more in class. I know that when it came time to finally explicitly code in an object–oriented manner, there was confusion on what constituted on a getter/setter.
Projects were a bit confusing to get started sometimes. More hints on the setup for some of the projects (or customized workflow for diff projects) could be helpful.
The lack of ability to take notes on a laptop during class severely limited by learning ability. I understand the reason for the rule, but it was definitely detrimental for me personally. People that are going to distract themselves with their laptop aren't going to pay attention, even if they're forced to turn them off. I think the rule is really just detrimental to student's abilities to take notes during class more than anything.
Honestly, the quizzes were the hardest part. The time was a bit short, and sometimes I could not get the answers in 3 minutes.
I think the aspect of the course that I found most challenging is the projects, since there was really little documentation on the desired output format in hackerrank, leading to a lot of debugging when it doesn't pass the hidden testcases. In the future, I think more directions could be provided in order to prevent this from happening.
The course material builds on each other, so towards the end of the semester, it becomes harder to remember and apply the material from earlier in the semester.
N/A
I found all of the project specs a bit extra, although I think they're good for those who need to learn. One piece of advice would be to not require the git log or the git SHA in the readme because I would always have to re–write the file when I was finishing the assignment.
Exercises were challenging because unknown syntax would sometimes pop up. Maybe that could be explained beforehand? It would save a lot of time spent raising hands.
I think the repetition can be dialed down and replaced with more content. While I think they are helpful when some material is challenging, I also think that there was a lot more to be learned that I was eager to hear about. I think the repetition could be added elsewhere besides in the lecture, which may increase the difficulty of the class, but I think it would be more rewarding. Also, there were several times where I wanted to ask a question, but I wasn't seen; I was a bit shy, so it was hard for me to call out that I had questions to ask. The 1 on 1 cold call format felt like it limited when I could ask questions, so I think shorter cold calls might help.