SEVERAL QUESTIONS WHICH WORK FOR ALMOST ANY COMPUTER SCIENCE EXAM

Barbara M. Anthony
Department of Mathematics and Computer Science
Southwestern University
Georgetown, TX 78626
512-863-1448
anthonyb@southwestern.edu

ABSTRACT

Many computer science courses use final exams as one means of evaluating student performance within the course, but developing meaningful assessments can be both difficult and time-consuming. Students and faculty alike can experience both stress and frustration with various aspects of the exam process. By stepping back and evaluating the overall goals of computer science courses and the curriculum, some questions arise as viable options for final exams in a variety of computer science courses. These questions enable students to consider prior courses, projects or presentations within the course, future use of the material, or handle what may be perceived as incomplete coverage in the final exam. Some potential concerns about these questions are addressed. While these questions will not replace all of the traditional questions, they may provide different types of assessment information, and enable reflection for both students and faculty.

INTRODUCTION

Based upon the prevalence of final exams in computer science courses, faculty value them as a means of assessment, but "preparing a written exam that truly and deeply tests students' knowledge in theoretical material is difficult and time-consuming" [7]. Existing research in computer science education considers options including automatically assessed electronic exams [10], collaborative group tests [5], concept questions [13], exam wrappers [11], and oral exams [7], yet all have their own challenges and limitations. While some questions permit minor adaptations that allow meaningful re-use, others are best not used again, due in part to the exam repositories widely known to exist in various forms on many campuses and the Internet.

These challenges perhaps lead instructors back to the question of what are the goals of exams, and how are they used. Are they for departmental or university-wide assessment of student learning outcomes? Are they a test of knowledge learned, a motivator for students to review material, an exercise in memorization, a demonstration of knowledge transfer, or a chance for reflection? Should exams challenge students, or allow them to feel a sense of accomplishment? One study reviewing questions on multiple introductory computer science exams observed that many programming-related questions required students to write code and incorporate numerous programming concepts into a single question, leading them to posit "that these types of questions are one reason why a significant fraction of CS1 students perform poorly or fail to complete the course'' [9].

ADDRESSING CONCERNS

Before proposing several questions that have worked for one instructor in a variety of courses to allow students to demonstrate understanding or knowledge gained, there are first some natural concerns to address. Many publishers have provided test banks associated with textbooks for years, so clearly a list of exam questions does not result in quick and easy production of final exams that universally meet the goals of instructors. These proposed questions would typically only comprise a small portion of a final exam, but may provide a complement to common problem-set or coding questions. It is quite likely that different variants of the questions will be used based upon the setting, perhaps even including institutional culture. For example, sometimes an instructor may need to provide more guidance about what is considered an 'exam-worthy' question. But if the expectation of doing so has been established throughout the semester, or in prior courses taken by all students, fewer details may be needed.

Should instructors be concerned that students may know these questions ahead of time? If all students have access to the same set of exam questions ahead of time, and all students and the instructor know this, then many concerns related to equity are alleviated. A goal, then, may be to ensure that questions still provide meaningful information about what students have learned. Yet even having the questions in advance does not guarantee uniform student performance. When students in an introductory programming course wrote practice exam questions that were available to all, those who wrote the questions did significantly better [6]. Likewise, a common strategy for instructors in disciplines where essay exams are more prevalent is to give students a certain number of essay questions in advance, and then pick a subset of them as the questions on the final exam. They may or may not have additional questions; computer science exams with these reflection-style questions are highly unlikely to consist solely of these proposed question. Ideally, of course, exam questions are not fully a surprise to students, but rather a chance to demonstrate their knowledge learned in the course. Furthermore, two-stage exams used as a learning experience for students have shown some success in computer science [12].

QUESTIONS, RATIONALE, AND DISCUSSION

The following four types of questions aim to make a final exam not merely a closing note on a semester-long course, but rather to remind students that exams are part of the learning experience, that courses are not taught in isolation, and that the questions placed on the exam reflect, to some extent, the choices of the exam creators. The amount of guidance needed to be included in the question may vary with the class, the experience of the students, and the culture of the institution. Students are very unlikely to give substantial but identical answers to these questions. Slight variations in the questions may mean that if students are aware of the likelihood of these types of questions, that if they prepare for a number of the likely variants, they are preparing for the exam in exactly the way that many instructors hope they would.

Write Your Own Exam-Worthy Question

Many students legitimately put forth effort in preparing for final exams. Unsurprisingly, exams are stressful for students, who have ‘’crammed as much as they possibly can and [are] hoping that the material they have studied would indeed be on the exams” [12]. While exams are likely to always have some amount of stress associated with them, perhaps beneficially so, exam creators may at least be able to make students feel like their preparation was not for nought. Accordingly, there can be an exam question that asks students to *Write and solve a question on material covered in this course that was not covered in this exam.* Often it is helpful to give students guidance about difficulty, perhaps saying at least as difficult as the average difficulty on the exam, or giving a suggested number of points that it is worth, in comparison to other exam problems. Likewise, indicate if the question can be one previously encountered in this course, or if students are expected to make up their own problem, and if so, how substantially or trivially it must differ from previously seen material. Sometimes it may be appropriate to reward students for creativity or applications to particular domains.

This type of question may be most successful if students have previously written questions about course content; that practice can be a valuable in-class or out-of-class activity. These questions can also provide insight into what students studied, though that may be because it was interesting or relevant, or because it posed particular challenges. Though inclusive pedagogy goes far beyond relatable examples, students may also use this question to connect course content to topics which are personally relevant. Instructors may also be able to use questions which students create in later semesters. In addition, since students are likely to choose a question in which they are confident about the answer, mistakes in the solutions students pose to their own questions may reveal different kinds of gaps in student learning.

Relate The Course To A Prerequisite

Though computer science curricula differ, most computer science majors are substantially sequenced, with one or more prerequisites often appearing in the representative syllabi in the Computer Science Curricular 2013 [1]. These prerequisites are often put into place for good reason, perhaps due to particular content, or exposure to certain programming languages or tools. Yet at times prerequisites remain for historical reasons, or perhaps well-intentioned notions that experience in one course will help in another, which may be true, but not sufficient to merit prerequisite status when considering other scheduling constraints and restrictions. If prerequisites are needed, generally the reasons should be apparent and logical to students as well as faculty.

A question can be more general, asking students to pick and describe a concept from a prerequisite course that they found useful, or even simply asking them to explain why the prerequisite is in fact appropriate for that course. Yet the question could be more specific, asking students in a data structures class who had experience with Java’s ArrayLists in CS1 to explain not only the underlying data structure, but also why encapsulation of the details of the implementation allowed them to effectively use an ArrayList in CS1. Care should be taken with questions of this nature to ensure that transfer students or students with credit by examination, for example, are not negatively impacted from not having taken the course at the institution.

While it may be tempting to ask why a current course is a prerequisite for a future course, that question likely requires speculation on the part of students.

Presentation, Project, Or Activity Connections

Collaborative assignments and projects are one of a number of high-impact educational practices that have been shown to be effective, and notably for students from a variety of backgrounds [8]. Thus, many faculty have incorporated these types of activities into their courses [2,3], often using in-person class meetings for group project work or presentations. Admittedly, some instructors have concerns about using class time for presentations because of the potential for students who are not presenting to be disengaged. Yet if students are told that there will be an exam question about the presentations, attentiveness can be increased. Some questions can allow the students to pick which presentation or project they want to report on, asking *Which presentation, other than the one you were involved in, most made you want to learn more about the topic? Provide specific details from the presentation to support your answer*. Certainly, a student could pay close attention to only one presentation, and then sculpt their answer to fit the prompt. Thus, the instructor could impose additional requirements about the selected presentation, ask for comparisons between multiple presentations, or constrain the additional details. With all of these options, care must be taken to not disadvantage students inappropriately, including students who may have had excused absences for one or more presentations. If students are required to include the names of the presenters, ensure that they have had ample opportunities to learn the names, and realize that this is an important detail. Beyond the context of the course, knowing names can have professional benefits as well.

Activities, whether completed in the class or as outside assignments, can also provide exam fodder and feedback about the course. Playing cards, pop beads, and nesting cups are manipulatives useful in solidifying the understanding of data structures. Students could be asked what manipulative best illustrated a concept for them, and to explain why, which requires a detailed answer, rather than just recalling which items were used. Alternatively, knowing which manipulative they found least helpful can also be telling, and can signal to students that truthful answers are valued, rather than having students try to guess what they think the instructor wants to hear. Again, the option exists for the instructor to specify the activity rather than have students choose which they discuss.

Twenty Years From Now

With the ability to look up vast quantities of information on the Internet, students can readily look up many of the facts learned in college, and thus do not need to commit the majority of them to lifelong memory. Yet higher education still provides value in terms of analyzing those facts. One podcast episode recognizes that many in a calculus class will not remember specific details about the content, but asks how the lessons you learn in a particular class will continue to be impactful twenty years later [4]. That question can be asked of educators in terms of how they develop their course, but can also be asked of students as they complete a course, namely *What do you hope you will remember from this class 20 years from now, and why?* That question may be even more apt for a discipline like computer science, with many aspects of the discipline changing rapidly, and faculty frequently teaching in languages that did not exist when

they were in college. It is important for students to have an understanding of what are the core principles that will not change, what are the areas where the details change but the big picture stays the same, and what will require the most work to stay current.

Again, care needs to be taken for students to answer the question asked, rather than what they think the instructor wants to hear. It may be appropriate to indicate in the question that a student could receive full credit if they say they will remember nothing, provided they justify it. For an upper-level majors course, the question may be framed in terms of what a practicing software engineer (or other appropriate profession) would ideally remember. For an introductory course, the question may involve asking students to indicate their professional goals for the future, and what from the course they will incorporate. Students are often brutally honest in the questions about what they will remember, and while it can be painful to hear that a student felt whole sections were irrelevant, that is also valuable feedback often not reflected in course evaluations. Such insights can lead to framing the material differently, or, on occasion, considering its appropriateness and relevance.

CONCLUSIONS

The author has used variants of all of these questions not only in computer science courses from the introductory level to the capstone course, but also in general education courses, and has found them to be valuable not only in assessing student performance, but also in considering the effectiveness and organization of the courses. Colleagues at multiple institutions have responded favorably to the idea of including these questions, with some later reporting they have adopted the ideas themselves, and suggested it to others. Some students have commented favorably upon the inclusion of said exams, and future work may include more rigorous study of performance on these questions as compared to other aspects of this course. Choosing one or two of these questions to incorporate, either as written or a variant, may help an instructor start writing an exam that allows students and teachers to benefit from the examination process.

REFERENCES

[1] ACM/IEEE-CS Joint Task Force on Computing Curricula, *Computer Science Curricula 2013*, ACM Press and IEEE Computer Society Press, 2013.

[2] Anewalt, K., Polack-Wahl, J., Beidler, J., Smarkusky, D., Group projects across the curriculum, *Journal of Computing Sciences in Colleges,* 19 (2), 232-237, 2003.

[3] Anthony, B., Operations research: broadening computer science in a liberal arts college, *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12),* 463-468, 2012.

[4] Burger, E., Stayton, J., Higher Ed: The '20-Year' Education Question, *KUT Higher Ed [Audio podcast]* released October 11, 2015, http://kut.org/post/higher-ed-20-year-education-question, retrieved May 22, 2017.

[5] Cao, Y., Porter, L., Evaluating Student Learning from Collaborative Group Tests in Introductory Computing, *Proceedings of the 48th ACM Technical Symposium on Computer Science Education (SIGCSE '17),* 99-104, 2017.

[6] Denny, P., Generating Practice Questions as a Preparation Strategy for Introductory Programming Exams, *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15),* 278-283, 2015.

[7] Gharibyan, H., Assessing students' knowledge: oral exams vs. written tests, *Proceedings of the 10th Conference on Innovation and Technology in Computer Science Education* (ITiCSE '05), 143-147, 2005.

[8] Kuh, G., *High-Impact educational practices, what they are, who has access to them, and why they matter,* Washington, D.C.: Association of American Colleges and Universities, 2008.

[9] Petersen, A., Craig, M., Zingaro, D., Reviewing CS1 exam question content. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE '11)*, 631-636, 2011.

[10] Rajala, T., Kalia, E., Lindén, R., Kurvinen, E., Lokkila, E., Laakso, M.-J., Salakoski, T., Automatically assessed electronic exams in programming courses, *Proceedings of the Australasian Computer Science Week Multiconference (ACSW '16)*, Article 11, 2016.

[11] Stephenson, B., Craig, M., Zingaro, D., Horton, D., Heap, D., Huynh, E., Exam Wrappers: Not a Silver Bullet. *Proceedings of the 48th ACM Technical Symposium on Computer Science Education (SIGCSE '17),* 573-578, 2017.

[12] Yu, B., Tsiknis, G., Allen, M., Turning exams into a learning experience, *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (SIGCSE '10), 336-340, 2010.

[13] Zingaro, D., Petersen, A., Craig, M., Stepping up to integrative questions on CS1 exams. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12),* 253-258, 2012.