Experiences Teaching a CS1 Common Course across 7 Institutions

Frank Vahid
Computer Science and Engineering
University of California, Riverside
Riverside, California, USA
vahid@cs.ucr.edu
Also with zyBooks

Ashley Pang
Computer Science and Engineering
University of California, Riverside
Riverside, California, USA
apang024@ucr.edu

ABSTRACT

We describe our experience organizing and teaching a CS1 "common course" pilot across 7 institutions during the 2023 spring term. A common course is a comprehensive centrally-designed course taught nearly identically across multiple institutions. It goes beyond common inter-institution sharing of ideas and resources, and instead is essentially the same course taught by different instructors, akin to multiple coordinated sections of a course at one institution. We describe the experience of 7 instructors who voluntarily joined the pilot, which included instructors at 5 state universities and 2 community colleges. The common course's comprehensive design included a 15-week configured CS1 C++ online zyBook having weekly interactive readings, coding homeworks, programming assignments, and quizzes (all auto-graded); a midterm and final exam; a syllabus with schedule, grade weights, policies (late policies, cheating policies, etc.); support for teaching active lectures including detailed lecture notes and coding examples; and bi-weekly meetings among the 7 instructors plus an informal shared TA. Overall, the courses went smoothly for students, and the instructors all strongly indicated they benefited from the experience, and would do it again and recommend it to others. They listed key benefits to include time savings (which freed them to perform higher-value, more enjoyable tasks), state-of-the-art tools and pedagogy (like auto-grading and active lectures), the coding examples in the lecture notes, the ability to compare their students' performance to others, and the camaraderie and idea exchanges at the bi-weekly meetings.

CCS CONCEPTS

• Social and professional topics - Professional topics - Computing education - Computing education programs - Computer science education - CS1

SIGCSE 2024, March 20–23, 2024, Portland, OR, USA.

ACM ISBN 979-8-4007-0423-9/24/03.
https://doi.org/10.1145/3626252.3630847

KEYWORDS: CS1, teaching, programming, common course, master course, exemplar course

ACM Reference format:

Frank Vahid and Ashley Pang. 2024. Experiences Teaching a CS1 Common Course across 7 Institutions. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024), March 20-23, 2024,* Portland, OR, USA. ACM, New York, NY, USA. 7 pages. https://doi.org/10.1145/3626252.3630847

1 INTRODUCTION

Introductory programming courses (CS1) and other courses often use shared resources. For example, many CS1 courses use one of a few widely-used textbooks, and instructors often use publisher-provided lecture slides or notes, pre-designed homework problems (end-of-chapter or via an online homework system with autograding), pre-designed programming assignments (from an auto-grading system or online repository), test bank questions, online videos, and more. Instructors may use online repositories or tools like Nifty assignments [1], Parson's problems [2], or Coding Bats [3]. However, ultimately each instructor (or school) still designs and runs their own course.

In October of 2021, we proposed via a post on the SIGCSE mailing list [4] the idea of a "master class" or "core class" -- what we here will call a "common course" -- for introductory CS courses. The proposal was as follows:

There's a [common course] that's been fully designed, including the syllabus, readings, homeworks, quizzes, exams, in-class examples, etc. [Common courses] would incorporate known best practices so students everywhere benefit, not just those with instructors who know and can implement best practices. Due to economy of scale, items used in class (e.g., programming assignments, exams) could be extensively developed (e.g., randomized problems), widely tested, and analyzed for continual improvement. Nearly everything would be auto-graded. The instructor's contribution is thus elevated to adding value to that core, via doing examples in class, explaining tough concepts, answering questions, providing motivation, giving help, facilitating peer instruction, training/managing TAs/tutors, etc. Variations would thus be in those value-added features, and ideally research and dissemination would help us become better at that value-added part.

The post used the analogy of how multiple sections of a course are taught at many large universities, where the course is centrally designed, and then different instructors teach various sections that are mostly synchronized and share exams, resources, etc.

What ensued was a lively discussion, with some agreeing, but others warning that such courses would diminish the role of instructors to mere curators and/or supervisors.

In Fall 2022, we put out a call for participants for a common course pilot in order to answer the question "Does the use of a common course diminish, or does it elevate, the role of instructors?" We centrally designed a CS1 course in C++, and 7 selected instructors then taught that course at their respective schools in Spring 2023. This paper details that experience. The strong conclusion from the 7 instructors is that the common course was an excellent experience for the instructors with numerous substantial benefits, and they felt the students had a good experience. There was no sense of their role being diminished.

2 COMMON COURSE IDEA

In this section, we define our view of a common course. Nearly the entire course is pre-designed by course maintainers, ideally incorporating many CS education best practices and automation. A common course's pre-designed elements are shown in Figure 1.

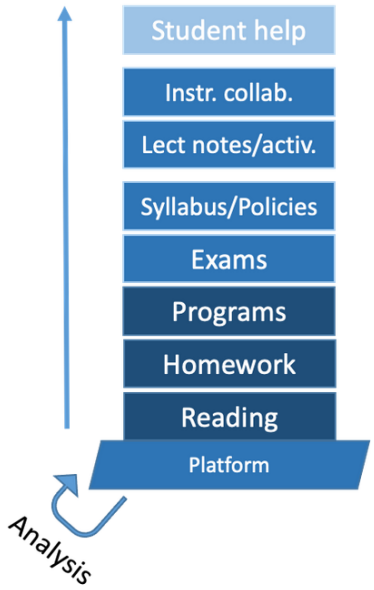

Figure 1: Elements pre-designed in a common course. The dark blue items are commonly mostly-completed in many courses, the lighter blue are relatively new in the common course, and the lightest blue (student help) is ideally included but wasn't in our pilot.

The course exists on a cloud platform, so instructors are freed from platform selection and setup. The weekly readings are pre-selected, and weekly homeworks and programming assignments are provided, all auto-graded. Those items are shown darker because in fact hundreds of schools already adopt such items from publishers like zyBooks (Wiley), Cengage, or Pearson, or from open-education resources like Runestone [5, 6, 7, 8], though usually requiring more configuration than in the common course. Additionally, a common course includes exams, a syllabus with policies, and detailed lecture notes with examples and activities, and also has instructors meeting regularly and collaborating. Those are shown in lighter blue because usually those are mostly designed/done by instructors (perhaps derived from samples or test bank questions), but are provided off-the-shelf in the common course. Finally, a common course may include ways for students to get help, such as via discussion forums (perhaps even shared among schools), common teaching assistant resources, or artificial intelligence tools. Those are shown in light blue because we did not implement such help in our experiment. The instructor steps in and provides lectures, motivation, help, etc., but is freed from most aspects of designing the course such as choosing programming assignments, creating exams, defining policies, etc. Ideally, but not essentially, the instructor is teaching the common course in sync (or close) with colleagues.

Upon seeing all that comes pre-designed in a common course, many would say "But that's the instructor's job" or "You've replaced the instructor almost entirely." Indeed, much of an instructor's time today is spent on designing/selecting homework and programming assignments, creating exams, preparing lecture notes, etc. However, a key question is whether pre-designing those items *diminishes* an instructor's role, or instead *elevates* their role -- something this pilot sought to explore.

A potential benefit is that a common course can incorporate various CS education best practices, such as active lectures, heavily scaffolded learning, auto-grading, equitable policies, and more, meaning the instructor can benefit from those practices without themselves having had to research those practices and then figure out how to design them into their course, yielding a higher-quality more-robust course. Another key benefit is instructor time savings, which can be used to add value to the course via more individualized help, for example. Another benefit is the ability of the common course maintainers to do more data analysis than possible for instructors due to economy of scale, potentially sharing tasks among common course instructors and/or staff, yielding continual improvement in the quality of exams, programming assignments, etc. Instructors can also benefit from an opportunity to interact with and learn from fellow instructors, which at many institutions doesn't happen due to only one instructor teaching a given course. And, instructors can get some sense of how their students are doing relative to students at other schools, which today is challenging. A drawback is less flexibility in course design, such as the ordering of topics, the number of quizzes/exams, or the difficulty level of assignments or exams.

We note that the Gates Foundation is working towards what they call "Exemplar Courses", having a related theme -- carefully-designed courses that institutions can adopt, in their case having an emphasis on equitable treatment of students [9]. A key difference from our pilot seems to be that instructors can modify Exemplar courses substantially. Their initial focus is introductory

statistics and chemistry. We are unaware of efforts to create a common course for CS1 or for other CS courses, beyond perhaps a couple campuses sharing a course, such as a community college feeding into a university, or cooperating campuses in a larger university system. Of course, campuses articulate their courses for transfer purposes. Many people have taken inventories of commonly taught topics in CS1 courses [10, 11] or covered in CS1 exams [12]. Standardization of courses is more common for high-school AP CS courses [13, 14, 15].

3 THE COMMON COURSE PARTICIPANTS

An email was sent to several hundred CS1 instructors across the U.S. in Fall 2022 describing the common course and inviting applications to participate in a Spring 2023 common course offering. The invitation explained the concept of a common course, and provided a sample syllabus plus the table of contents for a 15-week pre-configured zyBook, which included weekly readings, auto-graded coding homework, and auto-graded programming assignments, in C++, Java, and Python. The invitation also explained what it would mean to participate, namely that each participating instructor should strive to closely match the common course's schedule, policies, and practices, use the common course's quizzes, midterm, and final exam, meet a few times during the term, and participate in a few instructor and student surveys. Based on the 32 applications received, we decided to proceed with a C++ offering, as we had to pick one language for the pilot, C++ was the course we had the most experience with, and we had a sufficient number of applications (10) for C++. We selected 7 instructors based largely on their stated ability to meet the requirements, especially regarding topic selection in the 15-week configuration.

The 7 instructors all taught college-level CS1 in C++, and had taught for at least several years, with at least three having over a decade of CS1 experience. The schools included two community colleges, and five public state universities, with one typically ranked as a top-40 CS department. One university course was an online course. Another primarily served high-school students in a special enrollment program. Most course sizes were in the 20-40 students range, with one course having just over 100 students. 5 courses had a 15-week term, 1 had a 16-week term, and 1 had a 10-week term due to being on the quarter system (thus not covering the last few weeks of the common course's topics). All the semester schools started in January and ended in May, with 1 school being a week ahead and 1 being a week behind the other 4 that were mostly in sync. The quarter school started in March and ended in June, but the instructor attended all the common course meetings during the semester.

4 THE COMMON COURSE DESIGN

We designed the common course to be based largely on previous offerings of our introductory C++ course, which was mostly based on a zyBook. We participated in the experiment as one of the 7 instructors and as an informal teaching assistant. A zyBook has interactive readings with points awarded for participation, and built-in auto-graded homework problems (C++ code reading and writing in this case), shown by various researchers to improve student grades [16, 17, 18, 19]. The zyBook also had built-in programming assignments using the zyLab system. The course had gone through a decade of continual improvement of the zyBook configuration, exams, policies, etc. That instructor, along with two colleagues and some Ph.D. students, spent some time adapting that course into one that could be offered by other schools. Such adaptation included extensive analysis of exam questions (multiple choice, and code writing problems) for accuracy and validity, and modifying questions as appropriate. The adaptation also included getting data from zyBooks on the most common weekly configurations of their C++ content for 15-week semesters, and making modest refinements to the course contents to match those common configurations.

Prior to Spring 2023, the common course instructors were shown a 15-week preconfigured zyBook and asked to provide feedback. Based on that feedback, some topics were added, removed, or re-ordered, and some programming assignments were refined as well, yielding a base zyBook shown in Figure 2.

Figure 2: 15-week pre-configured zyBook contents.

The weekly topics included: input/output, variables/assignments (2 weeks), branches, strings, loops (2 wks), the midterm followed by vectors (C++'s version of arrays), user-defined functions (2 wks), streams and files (1/2 wk), objects and classes (1.5 wks), and finally pointers (2 wks). The resulting pre-configured zyBook had 15 chapters (one per week), with about 10-15 sections per week. The zyBook is heavily scaffolded, where each section included "reading" known as Participation Activities (PAs) in zyBooks, involving text, typically 1-2 animations, and about 15-20

interactive learning questions with immediate automated feedback including explanations for right and wrong answers. Each section typically had 3-5 "homework" problems known as Challenge Activities (CAs), typically being auto-graded and often auto-generated code reading or code writing problems. Each chapter had ~5 auto-graded programming assignments, known as Lab Activities (LAs) in zyBooks, which provide immediate feedback and partial credit based on the number of test cases passed with unlimited attempts. Based on analyses of the course designer's own course's past student behaviors, the designer estimated weekly student time to be 3 hours for PAs, 3 hours for CAs, and 3 hours for LAs, totalling about 9 hours per week.

Once the course designer was satisfied, all 7 instructors cloned that base zyBook, to create 7 unique but nearly identical class zyBooks, one per school. The cloning approach meant that each instructor could only access data for their own students, essential for privacy reasons. Instructors could make small changes to their zyBook clone if desired, such as one instructor adding a few extra topics, another adding a larger programming assignment, and one adjusting the last weeks to stretch to 16 weeks rather than 15.

Item	Description
zyBook pre-configured for 15 weeks	Each week's chapter included about 20 animations and 150-200 learning questions (PAs), 30-40 small code reading/writing homework problems (CAs), and 5-7 programming assignments (LAs), all auto-graded with immediate feedback.
Syllabus	2 pages (optimized for conciseness), weekly deadlines (recurring pattern / rhythm), grading policies, late policies, collaboration policies, etc.
Forms	Google forms for requesting late exception, reporting grade discrepancies, or retracting a submitted program.
5 quizzes	Initially hidden, located in additional chapter in zyBook, at-home, unhidden about every 2-3 weeks, given 36 hours, programming activity in zyBook (LA), auto-graded, no limit on attempts
Midterm and final	Each half multiple choice half code writing. Sample midterm and final also provided. Initially provided as a google doc (later updated to QTI and zyLab).
Bi-weekly meetings	Instructors met online every two weeks for 1 hour, to get info and tips from the course designer, ask questions, share experiences and ideas, etc.
Informal TA	A Ph.D. student at the course designer's university acted as an informal TA, helping the instructors with tasks like setting up exams, helping the course designer prepare items like survey forms, etc.

Table 1: Key items provided to common course instructors.

Instructors were given additional items that formed the common course, summarized in Table 1. One item was a 2-page syllabus, optimized for conciseness and transparency to aid students, including course policies. The syllabus laid out weekly due dates, with PAs due Mondays, CAs due Fridays, and LAs due Sundays, all at 10 pm (not midnight, to encourage good sleep habits), yielding a helpful weekly rhythm. Grade breakdown was: PAs 10%, CAs 10%, LAs 20%, Participation 5%, Quizzes 5%, Midterm 25%, Final 25%. The late policy was no lates were accepted, but exceptions could be requested, wherein students complete work and then fill a form evaluated at end of term. The programming policy was all programming had to be done in the zyBook coding windows so time spent/effort could be analyzed and to reduce cheating because a student's programming history was visible to instructors. The collaboration policy allowed help but had to be documented in top-of-program comments and programs should primarily be the student's own work. More policies were included, such as reporting grade discrepancies using a form, retracting a program after it was submitted using another form, etc. All forms were provided as clonable Google Forms. The syllabus included encouragement and links to local help resources (which the instructors had to update), and discussed academic dishonesty as well. Instructors were requested to use this syllabus as a base and to modify as necessary by their schools, as many schools have some items that must be included. Instructors were asked not to inform students that their course was a common course, to reduce potential cross-campus cheating.

The common course included 5 quizzes in a chapter near the end of the zyBook, set to hidden. The syllabus indicated the week when each quiz would be available to students. Instructors were told to unhide each quiz in the appropriate week of their term. The quizzes used zyBooks' program auto-grader which provides immediate feedback and partial credit based on the number of test cases passed. Students were given 36 hours to complete each quiz at-home, with unlimited attempts.

A few weeks into the semester, instructors were provided with the midterm exam, and a sample midterm exam similar to the midterm that instructors could share with students. Near the end of the semester, they were provided with a final exam and with a sample final exam too.

The instructors met every two weeks on a 1-hour video call. The calls included the common course designer and informal teaching assistant (TA) providing information and tips/advice, and the instructors sharing feedback and ideas with the course designer and with each other and/or making requests of the course designer and TA. These friendly meetings often went long, lasting up to 1.5 hours, due to the lively discussions.

In all aspects, the course designer was cautious regarding student privacy. Regarding FERPA [20], no student information covered by FERPA was shared among instructors, even though allowing instructors access to each others' zyBooks would have been easy. Regarding institutional review board (IRB) oversight, the pilot was careful not to carry out research involving the students themselves. Instead, the focus in this paper is on the instructor experience; as such, the pilot was in line with what a modern

instructor might do anyways in trying to incorporate CS education best practices and collaborate with colleagues across institutions, on a more active scale.

5 INSTRUCTOR FEEDBACK

Overall, the pilot went smoothly. Instructors indicated their students did well, with no shortcomings relative to past terms. Instructors were anonymously surveyed about their experience using a Google form. Responses are summarized in Table 2. 6 instructors completed the survey (the 7th had a family emergency that week). Most questions had 5 Likert-like choices (e.g., much less, slightly less, same, slightly more, much more); the table only shows choices with non-zero counts.

Summarizing, instructors indicated they saved time, the course had fewer bumps, students on average did better (but some slightly worse, possibly due to stricter policies or harder material), bi-weekly meetings with instructors were very useful, and lecture notes were very useful. Instructors would recommend that others participate, and would participate again.

Question topic	Responses
Weekly prep hours vs. previous term	2 same 2 slightly less 2 much less Stated time-savers: zyLabs auto-grader, zyLabs similarity checker, lecture notes, exam creation, lab creation
"Bumps" (errors, complaints, etc.) vs. previous term	2 about the same 4 much less
Student performance vs. previous term	2 slightly worse 2 about the same 2 much better
Regular meetings with instructors was useful to me	6 strongly agree Comments: "favorite part ... awesome colleagues ... great ideas .. gem of the experience", "liked being informed", "connection great ... great ideas ... best part", "I was already happy with my course ... but still learned so much", "great new ideas ... support"
Lecture notes were beneficial	1 neutral 5 strongly agree
Pre-made exams (real and sample) were beneficial	1 agree 5 strongly agree
The common course would benefit under-resourced instructors and adjuncts	1 agree 5 strongly agree
I would participate again	1 agree 5 strongly agree

Table 2: Survey results from instructors.

An open question asked about the biggest benefits. Responses included:

- "Collaboration with other instructors ... great ideas .. reinforced that what I'm doing is comparable to others"
- "So many ... biggest was vetted exams ... preconfigured zyBook organized by week .. lecture notes saved time ... meetings were helpful"
- "Support of other instructors and staff"
- "Refreshed energy after 30+ years teaching"
- "New ideas ... part of community of instructors ... all the available materials"
- "Basics taken care of ... I could focus on the more important and fun parts of teaching ... less lonely being part of group of teachers"

An open question asked what could be improved. Responses included labs with more application or more student creativity, more examples, and more instructor support for asynchronous online classes.

After the term, in June 2023, the instructors participated in an online panel attended by ~50 CS education practitioners (instructors, authors, publishers, etc.). The instructors described their experience and then answered questions. Common themes:

- The common course led to a more "student centered" approach, because instructors spent less time doing "grunt work" (making lesson plans, assignments, exams, grading) and instead could focus more on student needs.
- Working as part of a group was a huge benefit. Many instructors today work alone. Instructors learned ideas from each other, like pre/post-exam self-reflection surveys, using Kahoot for active classrooms, grading exams via carefully-designed rubrics, giving the multiple choice and code writing parts of exams separately, making announcements and answering class structural questions mid-class rather than at the start of class, giving students a chance to do mock technical interviews, and more.
- Working as part of a group also yielded helping each other, such as explaining how to import QTI exams into Canvas, sharing small issues students experienced with labs so other instructors could avoid those issues, having many eyes to proofread exams to ensure smooth exams, etc. Also, one class was ahead of the others, so issues could be reported early and improved in those others.
- Instructors thought that students benefited from the common course due to the transparency of course material being pre-selected and paced using zyBooks, helping students schedule their time wisely. They also felt students benefited from the provided sample exams, which the instructors normally didn't provide.

6 DISCUSSION

The instructor bi-weekly meetings ended up being one of the most valued aspects of the common course. Interestingly, the common course designer originally only planned 2-3 meetings for the entire

term, not wanting to impose on instructors' limited time. But after a pre-term meeting and week 2 meeting, the instructors requested continued meetings, leading to bi-weekly meetings. In the future, when common courses have more participants, meetings may use subgroups to maintain quality interaction.

The lecture notes with examples were originally not part of the common course. But in the pre-term meeting, the instructors asked for more lecture content, so the common course designer converted their informal notes into more detailed notes for use by others; the instructors asked for those notes throughout the term.

The common course provided exams to instructors as multiple choice and code writing questions in a google doc, assuming instructors would then provide the exam to students however they were comfortable (online, bubble sheets, paper, etc.). We found that instructors wanted more, with many asking for QTI format to import into their learning management system like Canvas, Blackboard, and D2L. Our informal TA provided such QTIs. Some instructors wanted the code writing problems captured in zyLabs for auto-grading, which the TA also created. Overall, we found our assumption of the sufficiency of google docs to be incorrect. A more out-of-the-box solution to giving exams was widely desired. In the future, we may investigate providing exams in a cloud-based system like PraireLearn [21].

A threat to validity is the instructors self-selected to participate in the pilot, and thus were likely open to the common course idea. The experience of other instructors, especially if such instructors were forced to participate, could be quite different -- reluctant participants in nearly any activity can yield poor results from that activity. However, our goal here is not to explore the results of *forcing* instructors to participate in a common course. Instead, we believe that if early common course experiences are positive among willing participants, more instructors may open themselves up to willingly participating themselves.

A common misconception of a common course is that it is similar to a MOOC (massive open online course) [22, 23], where instructors create an online course typically with pre-recorded lecture video, homeworks, and perhaps some TAs, and then offer the course to thousands of students around the world. However, a common course is substantially different. Whereas a MOOC can operate without an instructor or with an instructor participating in a limited role, in a common course the instructor is essential. Rather than eliminating the instructor, a common course is designed to do the opposite: Free the instructor to provide *even more value* to their students, typically by giving the instructor more time for high-quality interactions with their students.

An often-expressed concern relates to academic freedom, namely that instructors should not be told how to teach their classes. Indeed, care must be taken not to squash creativity and allow exploration and progress. With that said, we note that many universities already prescribe to instructors many things related to teaching CS1 (textbook, syllabus, schedule, etc.). And, instructors may find that giving up some freedom (like subject ordering, exam design, late policy, etc.) actually yields more freedom at a more impactful level. In any case, progressing down a common course path should be done with care not to trample on instructors, and instead ideally would have willing participants. We also don't envision a single course for the entire country or world; rather, one common course for CS1 might be adopted by 50 schools for example, another by 100 schools, and so on. Those common courses may experiment with different techniques, "competing" with each other for participants, which may catalyze progress. And on this note, instructors who want to experiment should be free to do so, such as creating a CS1 targeting women's interests [24], or trying out grading for equity, etc. Even then, such instructors might start with a common course template and then make revisions; and if they publish improvements, those could even be incorporated back into the original common course.

A centrally-designed course offered by many instructors is not by itself new, as it is common at many large universities having tens of sections per course. Likewise, a centrally-designed course at multiple campuses is not new, as some multi-campus institutions do that already, such as a large community college system. What is unique here, and in some eyes somewhat controversial, is that these 7 instructors had no actual affiliation with each other, and in fact came from a wide diversity of different schools. Also, although 7 instructors might initially seem like a small pilot, one might consider that such a pilot has not to our knowledge been done before for CS1 across multiple institutions, so getting 7 unaffiliated instructors to agree to teach the same CS1 course may be viewed as a substantially new experience.

7 CONCLUSIONS

One vision of the future of CS1, and other college gateway courses like math or physics, is that more schools may start offering versions of a centrally-designed "common course" in order to reap numerous benefits like the use of modern automated tools, incorporation of proven pedagogical techniques, high-quality exams, freeing up instructor time for higher-value student interactions, and more. However, many express concern that such common courses may diminish the role of instructors. To help explore the question of whether common courses diminish instructors' roles, or instead enhance their role, we conducted a pilot. This pilot with 7 instructors from widely-varying institutions found no such diminishing. Instead, instructors agreed it was an outstanding experience, and recommended it for others. They especially expressed strong appreciation for the interactions with fellow CS instructors, which turned out to be an unexpected highly-desired feature of the common course. Based on this experience, we recommend that others experiment with common courses as well, to see if indeed they provide benefits and may become popular across schools.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant Nos. 2111323 and 2313793.

REFERENCES

[1] Nifty Assignments, http://nifty.stanford.edu/, accessed July 2023.
[2] Parsons Puzzles, https://computerscienced.co.uk/site/parsons-puzzles/.
[3] Coding Bats, https://codingbat.com/, accessed 2023.

[4] Vahid, F. Master classes in introductory CS? Post to the SIGCSE mailing list, October 13, 2021.
[5] zyBooks.com, accessed 2023.
[6] Cengage.com, accessed 2023.
[7] Pearson.com, accessed 2023.
[8] Runestone.academy, accessed 2023.
[9] Gates Foundation, Digital Learning Communications Brief, https://usprogram.gatesfoundation.org/-/media/usp/usp-what-we-do/pdfs/bmgf-gateway-course-communications_ext_030722.pdf, March 2022.
[10] Dale, N., 2005. Content and emphasis in CS1. ACM SIGCSE Bulletin, 37(4), pp.69-73.
[11] Caceffo, R., Wolfman, S., Booth, K.S. and Azevedo, R., 2016, February. Developing a computer science concept inventory for introductory programming. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (pp. 364-369).
[12] Petersen, A., Craig, M. and Zingaro, D., 2011, March. Reviewing CS1 exam question content. In Proceedings of the 42nd ACM technical symposium on Computer science education (pp. 631-636).
[13] CMU's CS1 curriculum, CS for All, https://www.csforall.org/members/carnegie_mellon_universitys_computer_science_academy/curriculum/cs1/, accessed 2023.
[14] AP CS Awesome, https://sites.google.com/view/csawesome/, accessed 2023.
[15] Ericson, B., Hoffman, B. and Rosato, J., 2020, October. CSAwesome: AP CSA curriculum and professional development (practical report). In Proceedings of the 15th Workshop on Primary and Secondary Computing Education (pp. 1-6).
[16] Edgcomb, A.D. and Vahid, F., 2014, June. Effectiveness of online textbooks vs. interactive web-native content. In 2014 ASEE Annual Conference & Exposition (pp. 24-460).
[17] Edgcomb, A.D., Vahid, F., Lysecky, R., Knoesen, A., Amirtharajah, R. and Dorf, M.L., 2015, June. Student performance improvement using interactive textbooks: A three-university cross-semester analysis. In 2015 ASEE Annual Conference & Exposition (pp. 26-1423).
[18] Irani, S. and Denaro, K., 2020, February. Incorporating active learning strategies and instructor presence into an online discrete mathematics class. In Proceedings of the 51st ACM Technical Symposium on Computer Science Education (pp. 1186-1192).
[19] McKinney, D., Edgcomb, A.D., Lysecky, R. and Vahid, F., 2020, June. Improving pass rates by switching from a passive to an active learning textbook in cs0. In 2020 ASEE Virtual Annual Conference
[20] Family Educational Rights and Privacy Act, https://www2.ed.gov/ferpa, accessed 2023.
[21] West, M., Herman, G.L. and Zilles, C., 2015, June. PrairieLearn: Mastery-based online problem solving with adaptive scoring and recommendations driven by machine learning. In 2015 ASEE Annual Conference & Exposition (pp. 26-1238).
[22] EdX, https://www.edx.org/, accessed July 2023.
[23] Coursera, https://www.coursera.org/, accessed 2023.
[24] Rich, L., Perry, H. and Guzdial, M., 2004. A CS1 course designed to address interests of women. Acm sigcse bulletin, 36(1), pp.190-194.