Using a Computer-based Testing Facility to Improve Student Learning in a Programming Languages and Compilers Course

Terence Nip*
University of Illinois at
Urbana-Champaign
Urbana, Illinois
nip2@illinois.edu

Elsa L. Gunter
University of Illinois at
Urbana-Champaign
Urbana, Illinois
egunter@illinois.edu

Geoffrey L. Herman
University of Illinois at
Urbana-Champaign
Urbana, Illinois
glherman@illinois.edu

Jason W. Morphew
University of Illinois at
Urbana-Champaign
Urbana, Illinois
jmorphe2@illinois.edu

Matthew West
University of Illinois at
Urbana-Champaign
Urbana, Illinois
mwest@illinois.edu

ABSTRACT

While most efforts to improve students' learning in computer science education have focused on designing new pedagogies or tools, comparatively little research has focused on redesigning examinations to improve students' learning. Cognitive science research, however, has robustly demonstrated that getting students to practice using their knowledge in testing environments can significantly improve learning through a phenomenon known as the testing effect. The testing effect has been shown to improve learning more than rehearsal strategies such as re-reading a textbook or re-watching lectures. In this paper, we present a quasi-experimental study to examine the effect of using frequent, automated examinations in an advanced computer science course, "Programming Languages and Compilers" (CS 421). In Fall 2014, students were given traditional paper-based exams, but in Fall 2015 a computer-based testing facility enabled the course to offer more frequent examinations while other aspects of the course were held constant. A comparison of 292 student scores across the two semesters revealed a significant change in the distribution of students' grades with fewer students failing the final examination, and proportionately more students now earning grades of B and C instead. This data suggests that focusing on redesigning the nature of examinations may indeed be a relatively untapped opportunity to improve students' learning.

CCS CONCEPTS

• **Social and professional topics → Computing education**; **Student assessment**; • **Software and its engineering → General programming languages**; **Compilers**;

KEYWORDS

Programming languages, compilers, testing effect, computer-based testing

ACM Reference format:
Terence Nip, Elsa L. Gunter, Geoffrey L. Herman, Jason W. Morphew, and Matthew West. 2018. Using a Computer-based Testing Facility to Improve Student Learning in a Programming Languages and Compilers Course. In *Proceedings of SIGCSE '18: The 49th ACM Technical Symposium on Computing Science Education, Baltimore, MD, USA, February 21–24, 2018 (SIGCSE '18),* 6 pages.
https://doi.org/10.1145/3159450.3159500

1 INTRODUCTION

While most efforts to improve students' learning in computer science education have focused on designing new pedagogies (e.g., pair programming [32] and CS unplugged [30]) or new pedagogical tools (e.g., block languages [36] and informative compiler error messages [28]), comparatively little research has focused on redesigning traditional models of assessment. The lack of research and development on efforts to change the way that we test students may mean that we are leaving valuable and viable options for improving students' learning untapped. One of the most robust findings in cognitive science, the testing effect, suggests that we can improve students by engaging them in more test-taking behaviors and discouraging them for from engaging only in rehearsal strategies such as rereading the textbook or rewatching a video lecture [3, 11, 15]. Unfortunately, weaker students are the ones most likely to persist in these less effective learning strategies [15].

Class sizes continue to grow at the University of Illinois at Urbana-Champaign, increasing the difficulty of assessing students in a fair and timely manner. In response to these difficulties, we recently created a engineering college-wide Computer-Based Testing Facility (CBTF) with the goal of reducing the overhead of administering examinations and thus enabling faculty to give more frequent

[*] Now at Google.

SIGCSE '18, February 21–24, 2018, Baltimore, MD, USA

ACM ISBN 978-1-4503-5103-4/18/02…$15.00
https://doi.org/10.1145/3159450.3159500

and smaller exams to their students. The CBTF provides a central testing facility with continual proctoring [39]. Rather than schedule a single hour for an exam with 100 or more students (and afterwards accommodate any students who cannot attend that hour due to time conflicts), each student selects an hour that is convenient for them during an exam window to take a proctored exam [37]. These exams have randomized content to discourage cheating and enable this asynchronous testing environment [7]. In Spring 2017, the CBTF ran 37,000 exams for 4,500 students in 18 courses from 5 engineering departments.

During Fall 2015, the enrollment of "Programming Languages and Compilers" nearly doubled from 104 students to 188 students. As enrollments had been increasing, the instructors were increasingly concerned that learning outcomes were declining and that failure rates were rising. From the instructors' perspective, students seemed to be relying too much on their peers or other resources when completing their machine problems (i.e., intensive coding problem sets) and were not actively engaging on their own. Consequently, the instructors, like many other instructors using the CBTF, increased the number of times that students were tested in a semester. In addition to the standard two midterms and a final examination, students were also tested four times on their understanding of their code for week-long machine problems. For the remainder of this paper, we treat this change in the course as a quasi-experimental study investigating the effect of using examinations as a learning tool. Most aspects of the course remained the same between semesters, having the same instructor and approximately the same assignments. The only major changes made were the shift from paper-based examinations to computer-based examinations, and the switch to using short examinations to assess students' efforts on and understanding of their machine problems. We evaluate the impact of these changes on students' learning by comparing students' performance on the final examination from both semesters. We explore the following research question, did switching to exam-based assessments for machine problems improve students' learning in programming languages and compilers?

2 BACKGROUND

Left to their own devices, students typically select study strategies that tend to be passive and focus on encoding processes such as rereading a textbook, reviewing notes, or rewatching lectures [15]. Students, especially those who are lowest performing, tend to select inefficient study strategies. Retrieval practice, often in the form of test taking, has been shown to produce better long-term retention in both clinical studies [9, 31] as well as secondary and university classrooms [1, 22, 25, 27], compared with restudying materials. For example, McDermott et al. [27] utilized a within-subjects design with middle school students where the course material was randomly assigned to be either tested, restudied, or not tested or restudied. Students recalled facts at a higher rate for course material that was tested than for course material that was either restudied or not tested. Similar results were found for factual recall with undergraduate students in an online Psychology course [1].

The benefits of retrieval practice, also known as the testing effect or test-enhanced learning, are likely due to the ways in which testing facilitates the representation and retrieval of information stored in memory. Successful retrieval is thought to change the information's representation in memory such that it becomes easier to retrieve in the future [2]. However, the beneficial effects of testing are also found for items that were answered unsuccessfully during initial testing, suggesting that merely engaging in retrieval attempts may potentiate future learning of material [29]. Kornell, Hays, and Bjork [16] suggest that retrieval attempts during testing facilitate deep processing of the material, strengthen pathways for correctly recalled information, and weaken pathways for information which was incorrectly recalled.

Although most research concerning the testing effect has used either identical questions or very similar questions as those used in the retrieval practice, a few studies have demonstrated improvements for rephrased questions [22]. Other studies have shown improved performance on new inferential questions covering previously tested material [4]. In addition, some studies have found that retrieval attempts enhances performance on related but untested material [6, 8, 19]. However, other studies have found no testing effect for related, but untested material [20, 38].

In the laboratory, much of the research concerning the testing effect has focused on memory tasks, while research in the classroom has utilized content focused on declarative memory, such as word pairs in second language learning [13], factual recall in psychology [24, 26], short answer questions in medical education [17], recalling facts from a lecture [5], and multiple choice questions involving recalling or applying definitions in a middle school science course [21]. The benefit of testing on problem-solving tasks, such as those found in computer science courses, is less clear. Some researchers have asserted that testing effects are lessened as the complexity of the information increases [10, 35]. However, other researchers have documented testing effects for more complex tasks such as reading comprehension and inference tasks, learning spatial relationships, and constructing concept maps [12, 14]. For example, McDaniel, Howard, and Einstein (2009) had undergraduate students read two passages explaining how pumps and brakes operate [23]. After either studying or engaging in retrieval practice, students completed a free-recall task, multiple-choice factual recall, and a short-answer inference task that asked them to apply their knowledge about brakes and pumps in novel ways. Students engaged in retrieval outperformed students who reread the material on all three measures. However, retrieval was not more effective than note taking for the inference task.

To our knowledge, only three studies have examined the benefits of testing in mathematical problem-solving contexts. Leahy, Hanham, and Sweller [18] engaged elementary students in solving problems involving reading a bus schedule. Students who engaged in repeated studying of worked examples outperformed those who studied an example and then completed practice problems on immediate post-test and performed the same as those who completed practice problems on a delayed post-test. Van Gog and Kester engaged novices in learning to solve problems involving electrical circuits by either studying four worked examples or studying two worked examples followed by attempting to solve two isomorphic problems [33]. Participants in the study-only condition scored the same as those in the testing condition on the immediate post-test and higher on the delayed post-test one week later. Van Gog et al. compared retrieval versus restudy in students engaged in learning problem solving from worked examples across four experiments and found no advantage for testing over repeated studying for problem-solving tasks involving electrical circuits or probability distributions [34]. However, these findings may be the result of samples with low prior knowledge and motivation to learn from the worked examples. In these studies, the participants had very little experience with the topic, indicated by their low pre-study conceptual scores. In addition, the authors do not indicate whether the participants received feedback from their retrieval attempts.

3 METHODS

CS 421, "Programming Languages and Compilers", is a large enrollment (100-200 students per semester) upper division course. Students learn about language design principles, abstract data types, functional programming, and type systems. Students also learn about the basics of lexing, parsing, syntax-directed translation, semantic analysis, and code generation.

In Fall 2014, 104 students took CS 421. The course administered 11 written homework assignments, 11 machine problems, 2 paper-based midterm examinations, and a paper-based final examination. Examinations were composed of short programming problems, a few computation problems, and a few multiple true-false questions. In Fall 2015, 188 students took CS 421. That semester, the course migrated to the CBTF while most other aspects of the course remained unchanged (e.g., students were still given 11 homework assignments). The content coverage and question structure of the midterm and final examinations were kept similar (e.g., true/false questions were kept the same, but students used graphical tools to draw parse trees rather than create them by hand). However, the course changed the way that four of the machine problems were submitted. Rather than turn in code at the end of a week of work on the problem, students were given a test in the CBTF that required the students to code one fifth of the machine problem. Which fifth the students completed was randomly selected by the testing environment in the CBTF.

As a quasi-experimental study, we argue that most aspects between the course were held constant. The instructor was the same both semesters. Students taking the course in Fall 2014 and Fall 2015 had taken similar prior coursework and had spent a similar number of semesters in the degree program. The number of assignments was held constant. Only the modality of the midterm and final examinations and the modality of assessment of four machine problems were changed. As we describe in the next following subsection, the instructors strove to maintain parity between semesters in how students were assessed on midterm and final examinations. Consequently, we believe that the switch to using the CBTF to test students' understanding of the machine problems constituted the primary treatment for students' learning.

This sequential study design has the added benefit of avoiding ethical dilemmas arising from randomly withholding the treatment from students and the logistical challenges of requiring a single instructor to run two different versions of the course in parallel. Although we could not use random assignment for the treatments, thus limiting the causal claims that we can make, the ecological validity, large sample size, and similarity of the students provide a sufficiently controlled and well-powered study to conclude that the findings from the study may be considered to be robust.

3.1 Details of the computer-based examinations

The testing environment for the CBTF relies on a web-based homework system called PrairieLearn. PrairieLearn is an open-source platform that provides native support for standard question types such as multiple-choice, short-answer, and multiple true-false. It also allows students to upload files such as text files or PDFs. Beyond these basic functions, the system also allows instructors to supply their own client- and server-side code to generate new question types.

When migrating the paper-based midterm and final examinations to the CBTF, the CS 421 instructors used many of the standard features in an attempt to mirror the content from the paper-based exams, asking students to complete multiple true-false questions, provide answers to computation problems in short-answer boxes, and had students upload text files to be graded manually by the course staff. The instructors also developed additional tools such as creating an in-browser code editor that enabled students to write their code within the PrairieLearn environment rather than write code in a separate program and upload it.

Further, the instructors developed two libraries using HTML and JavaScript: one to display proof trees (and have students be able to identify errors in the proofs) (Figure 1), and another to allow students to construct proof trees from scratch (Figure 2). Similarly, the instructors created a library that allowed students to "draw" parse trees in the browser so that students could demonstrate their mastery of grammars - in particular, recognizing when a grammar is ambiguous and how strings can be parsed given an unambiguous grammar. The instructors created the libraries so that students could perform similar tasks to what they performed on the paper-based examinations, maintaining as much parity as possible with examinations across semesters.

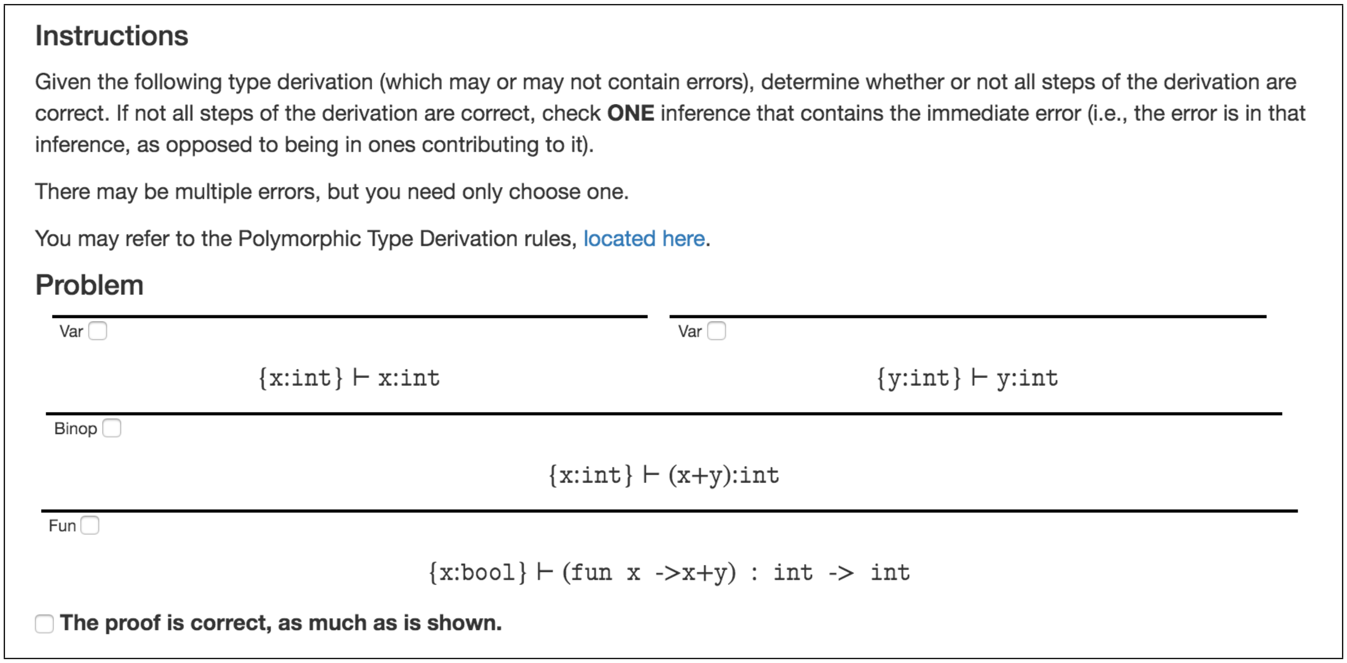
3.2 Analysis

To compare student performance across semesters, we compared student performance on the final examinations. The final examination is a stronger measure of student learning than course grades as there are fewer sources of variation in student grades in a single examination than over the course of semester (i.e., a student may get sick and fail to submit an assignment, dramatically reducing their grade even if the student learned the content). We perform an independent samples *t*-test to compare overall student performance on the final examination. We perform χ^2 tests to compare the distribution of students' grades on the final examination.

The statistical analysis reported in this paper was conducted by a educational psychology graduate student who is not affiliated with either CS 421 or the CBTF, minimizing bias in the reporting and interpretation of findings.

4 RESULTS

The mean final exam score was 4% higher in Fall 2015 (mean = 76.8%, sd = 15.6%) than in Fall 2014 (mean = 72.5%, sd = 18.6%). To determine whether altering the assessment schedule led to greater



Instructions

Given the following type derivation (which may or may not contain errors), determine whether or not all steps of the derivation are correct. If not all steps of the derivation are correct, check **ONE** inference that contains the immediate error (i.e., the error is in that inference, as opposed to being in ones contributing to it).

There may be multiple errors, but you need only choose one.

You may refer to the Polymorphic Type Derivation rules, located here.

Problem

Var □ {x:int} ⊢ x:int

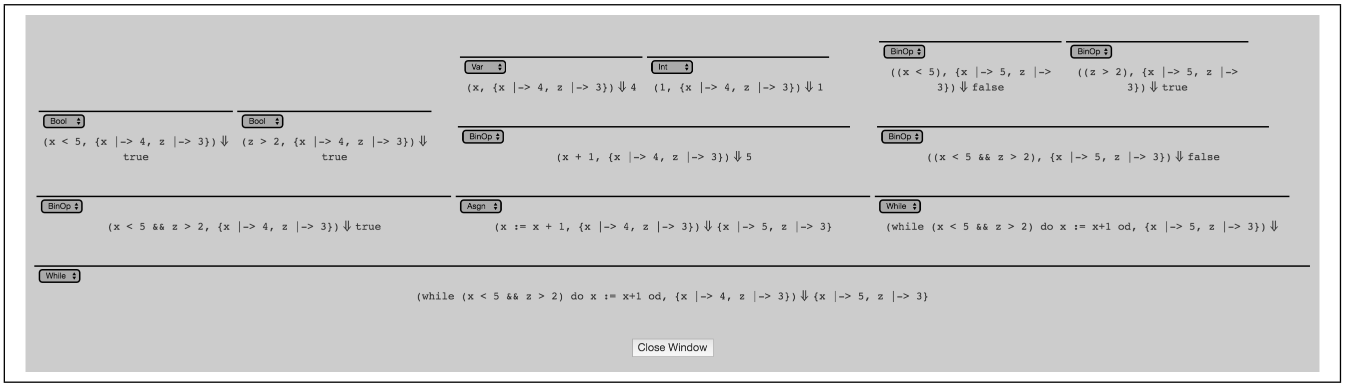
Var □ {y:int} ⊢ y:int

Binop □ {x:int} ⊢ (x+y):int

Fun □ {x:bool} ⊢ (fun x ->x+y) : int -> int

□ **The proof is correct, as much as is shown.**

Figure 1: A proof tree with checkboxes for identifying mistakes.



```
Var   (x, {x |-> 4, z |-> 3}) ⇓ 4
Int   (1, {x |-> 4, z |-> 3}) ⇓ 1
BinOp ((x < 5), {x |-> 5, z |-> 3}) ⇓ false
BinOp ((z > 2), {x |-> 5, z |-> 3}) ⇓ true
Bool  (x < 5, {x |-> 4, z |-> 3}) ⇓ true
Bool  (z > 2, {x |-> 4, z |-> 3}) ⇓ true
BinOp (x + 1, {x |-> 4, z |-> 3}) ⇓ 5
BinOp ((x < 5 && z > 2), {x |-> 5, z |-> 3}) ⇓ false
BinOp (x < 5 && z > 2, {x |-> 4, z |-> 3}) ⇓ true
Asgn  (x := x + 1, {x |-> 4, z |-> 3}) ⇓ {x |-> 5, z |-> 3}
While (while (x < 5 && z > 2) do x := x+1 od, {x |-> 5, z |-> 3}) ⇓
While (while (x < 5 && z > 2) do x := x+1 od, {x |-> 4, z |-> 3}) ⇓ {x |-> 5, z |-> 3}
```

Figure 2: An example complete proof tree derivation.

performance on the final exam, we conducted an independent samples t-test. The assumption of equal variances was tested with the folded F-test. The variance was different between the two semesters ($F' = 1.43$, $p = .04$), therefore a Satterthwaite correction was used. Shapiro-Wilk’s tests of normality indicated that the distribution of exam grades deviated from normality for both semesters. The central limit theorem suggests that independent samples t-tests are robust to deviations from normality with large sample sizes. Since this study employed large samples and the distributions were similarly negatively skewed, this test is appropriate for the data. The results indicate that students completing the final exam in Fall 2015 scored higher on the final exam than did students completing the final exam in Fall 2014 ($t(183.1) = 2.01$, $p = .046$) with a small effect size (d = .25) roughly equivalent to four-tenths of a letter grade (See Table 1).

Table 1: Comparison of mean scores on the final examination of Programming Languages and Compilers between Fall 2014 and Fall 2015

Fall 2014 mean (sd)	Fall 2015 mean (sd)	t-test	Effect Size Cohen’s d
72.5% (18.6%)	76.8% (15.6%)	$p = .046$	d = .25

Because the instructors were deeply concerned about increasing failure rates and students failing to achieve core learning objectives, we further examined the effect of the new testing regime on pass/fail rates in the course. The letter grade on the final exam for

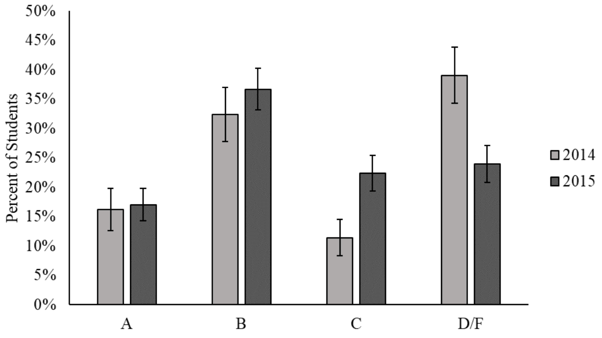


Figure 3: Final exam score distribution by letter grade.

each student was calculated and the grade distributions for the two semesters are shown in Figure 3. The grade distributions between the two semesters were statistically significantly different as measured by a chi-Square test of independence [$\chi^2(3) = 10.00$, $p = .02$]. The percentage decrease in the number of D/Fs (failing grades) is proportional to the increase in the number of B/Cs (passing grades).

5 DISCUSSION

This study primarily examined the effect of testing students on their machine problems rather than simply letting them submit their code after a week's worth of work. We found that this increased use of testing coincided with improved student performance on the final examination, indicating improved learning. In particular, we found that the distribution of grades was significantly different between semesters. After the increased use of testing, the percentage of failing grades (D/Fs) decreased proportionally with the increase in the percentage of low passing grades (B/Cs). These findings suggest that the use of increased testing primarily helped weaker students who may have otherwise failed the final examination. The stronger students appeared to be relatively unaffected, as the percentage of students earning A grades was not substantially different between semesters.

Alternate explanations of the findings include more lenient grading, an easier final examination, or an improved test-taking environment for students. Because the change to the CBTF was concurrent with the change in assessment philosophy for the machine problems, it is impossible to tease apart whether the improvement was environmental or due to the change in assessment strategy. We, however, argue that these explanations are weaker interpretations of the data. As mentioned in the methods sections, when making the switch to the CBTF, the instructors created their tools and examinations in an effort to maintain parity across semesters. The instructors actively sought to maintain the rigor of the examinations and went to great lengths to maintain similar modalities of testing in the CBTF. Additionally, a secondary motivation for the instructors to switch to the CBTF was to cope with the increasing scale of the course and to combat some of the ethical challenges that come with increasing class size (i.e., over-reliance of students on either their peers or external resources), in turn hopefully both stopping the decline in grades and motivating students to garner a greater understanding of the course material. The improvement in students' grades was rather much a surprise to the instructors, rather than a specifically sought after outcome biasing the study.

In contrast, the addition of an individualized test of understanding for the machine problems likely provided additional motivation for students to develop their own understanding of their code rather than overly rely on peers. Because students often rely on rehearsal strategies, such as reading someone else's code, they easily mistake familiarity with a solution for understanding of that solution. Requiring students to demonstrate their understanding in a test environment likely required students to develop their own understanding. This explanation is also compelling because we see that improvements in learning were primarily seen among weaker students (those earning failing grades). While the strong students (those earning As) likely were already learning the course content well before the change, the weaker students were now placed in a situation that aided their learning.

6 CONCLUSIONS AND FUTURE WORK

This study provides evidence that switching the assessment of students' understanding of an extended coding problem from simply turning in the code to testing their understanding of a portion of their code in an exam environment may improve students' learning. Critically, this switch significantly lowered the failure rate for students on the final examination. This finding is particularly exciting as the community continues to grapple with ways to improve retention rates and reduce failure in all core CS courses. Future studies will need to tease apart what effect a computer-based testing environment has relative to traditional paper-based examinations. Critically, these findings suggest that we should increasingly look to using the testing effect to improve students' learning in addition to our efforts to improve pedagogy and content.

ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation under Grants DUE 1347722 and CMMI 1150490. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] R. L. Bangert-Downs, J. A. Kulik, and C. Kulik. 1991. Effects of frequent classroom testing. *The Journal of Educational Research* 85 (1991), 89–99.
[2] R. A. Bjork. 1975. . Erlbaum, Hillsdale, NJ, Chapter Retrieval as a memory modifier: An interpretation of negative recency and related phenomena, pp. 123–144.
[3] P. C. Brown, H. L. Roediger, III, and M. A. McDaniel. 2014. *Make It Stick: The Science of Successful Learning.* Belknap Press.
[4] A. C. Butler. 2010. Repeated testing produced superior transfer of learning relative to repeated studying. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 36 (2010), 1118–1133. https://doi.org/10.1037/a0019902
[5] A. C. Butler and H. L. III. Roediger. 2007. Testing improves retention in a simulated classroom. *European Journal of Cognitive Psychology* 19 (2007), 514–527. https://doi.org/10.1080/09541440701326097
[6] J. C. K. Chan. 2010. Long-term effects of testing on the recall of nontested materials. *Memory* 18 (2010), 49–57. https://doi.org/10.1080/09658210903405737
[7] B. Chen, M. West, and C. Zilles. 2017. Do Performance Trends Suggest Widespread Collaborative Cheating on Asynchronous Exams?. In *Proceedings of the Fourth (2017) ACM Conference on Learning at Scale.* https://doi.org/10.1145/3051457.3051465
[8] M. T. H. Chi, P. J. Feltovich, and Glaser R. 2012. Using spacing to enhance diverse forms of learning: Review of recent research and implications for instruction. *Educational Psychology Review* 24 (2012), 369–378. https://doi.org/10.1007/s10648-012-9205-z

[9] C. F. Darley and B. B. Murdock. 1971. Effects of prior free recall testing on final recall and recognition. *Journal of Experimental Psychology* 91 (1971), 66–73. https://doi.org/10.1037/h0031836

[10] J. Hanham, W. Leahy, and J. Sweller. 2017. Cognitive load theory, element interactivity, and the testing and reverse testing effects. *Applied Cognitive Psychology* 31 (2017), 265–280. https://doi.org/10.1002/acp.3324

[11] M. K. Hartwig and J. Dunlosky. 2012. Study strategies of college students: Are self-testing and scheduling related to achievement? *Psychonomic Bulletin and Review* 19 (2012), 126–134. https://doi.org/10.3758/s13423-011-0181-y

[12] C. I. Johnson and R. E. Mayer. 2009. A testing effect with multimedia learning. *Journal of Educational Psychology* 101 (2009), 621–629. https://doi.org/10.1037/a0015183

[13] S. H. K. Kang, T. H. Gollan, and H. Pashler. 2013. DonâĂŹt just repeat after me: Retrieval practice is better than imitation for foreign vocabulary learning. *Psychonomic Bulletin & Review* 20 (2013), 1259–1265. https://doi.org/10.3758/s13423-013-0450-z

[14] J. D. Karpicke and W. R. Aue. 2015. The testing effect is alive and well with complex materials. *Educational Psychology Review* 27 (2015), 317–326. https://doi.org/10.1007/s10648-015-9309-3

[15] J. D. Karpicke, A. C. Butler, and H. L. Roediger. 2009. Metacognitive strategies in student learning: Do students practice retrieval when they study on their own? *Memory* 17 (2009), 471–479. https://doi.org/10.1080/09658210802647009

[16] N. Kornell, M. J. Hays, and R. Bjork. 2009. Unsuccessful Retrieval Attempts Enhance Subsequent Learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 35 (2009), 989–998. https://doi.org/10.1037/a0015729

[17] D. P. Larsen, A. C. Butler, and H. L. III. Roediger. 2009. Repeated testing improves long term retention relative to repeated study: A randomised controlled study. *Medical Education* 43 (2009), 1174–1181. https://doi.org/10.1111/j.1365-2923.2009.03518.x

[18] W. Leahy, J. Hanham, and J. Sweller. 2015. High element interactivity information during problem solving may lead to failure to obtain the testing effect. *Educational Psychology Review* 27 (2015), 291–304. https://doi.org/10.1007/s10648-015-9296-4

[19] J. L. Little, E. L. Bjork, R. A. Bjork, and G. Angello. 2012. Multiple-choice tests exonerated, at least of some charges: Fostering test-induced learning and avoiding test-induced forgetting. *Psychological Science* 23 (2012), 1337–1344. https://doi.org/10.1177/0956797612443370

[20] J. L. Little, B. C. Storm, and E. L. Bjork. 2011. The costs and benefits of testing text materials. *Memory* 19 (2011), 346–359. https://doi.org/10.1080/09658211.2011.569725

[21] M. A. McDaniel, P. K. Agarwal, B. J. Huesler, K. B. McDermott, and H. L. III. Roediger. 2011. Test-enhanced learning in a middle school science classroom: The effects of quiz frequency and placement. *Journal of Educational Psychology* 103 (2011), 399–414. https://doi.org/10.1037/a0021782

[22] M. A. McDaniel, J. L. Anderson, M. H. Derbish, and N. Morrisette. 2007. Testing the testing effect in the classroom. *European Journal of Cognitive Psychology* 19 (2007), 494–513. https://doi.org/10.1080/09541440701326154

[23] M. A. McDaniel, D. C. Howard, and G. O. Einstein. 2009. The read-recite-review study strategy: Effective and portable. *Psychological Science* 20 (2009), 516–522. https://doi.org/10.1111/j.1467-9280.2009.02325.x

[24] M. A. McDaniel, H. L. III. Roediger, and K. B. McDermott. 2007. Generalizing test-enhanced learning from the laboratory to the classroom. *Psychonomic Bulletin & Review* 14 (2007), 200–206. https://doi.org/10.3758/BF03194052

[25] M. A. McDaniel, R. C. Thomas, P. K. Agarwal, K. B. McDermott, and H. L. Roediger. 2013. Quizzing in middle school science: Successful transfer performance on classroom exams. *Applied Cognitive Psychology* 27 (2013), 360–372. https://doi.org/10.1002/acp.2914

[26] M. A. McDaniel, K. M. Wildman, and J. L. Anderson. 2012. Using quizzes to enhance summative-assessment performance in a web-based class: An experimental study. *Applied Research in Memory and Cognition* 1 (2012), 18–26. https://doi.org/10.1016/j.jarmac.2011.10.001

[27] K. B. McDermott, P. K. Agarwal, L. DâĂŹAntonio, H. L. III. Roediger, and M. A. McDaniel. 2014. Both multiple-choice and short-answer quizzes enhance later exam performance in middle and high school classes. *Journal of Experimental Psychology, Applied* 20 (2014), 3–21. https://doi.org/10.1037/xap0000004

[28] Raymond S. Pettit, John Homer, and Roger Gee. 2017. Do Enhanced Compiler Error Messages Help Students?: Results Inconclusive.. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 465–470. https://doi.org/10.1145/3017680.3017768

[29] L. E. Richland, L. S. Kao, and N. Kornell. 2013. Can unsuccessful tests enhance learning. *Educational Psychology Review* 25 (2013), 523–548. https://doi.org/10.1007/s10648-013-9240-4

[30] Brandon Rodriguez, Stephen Kennicutt, Cyndi Rader, and Tracy Camp. 2017. Assessing Computational Thinking in CS Unplugged Activities. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. ACM, New York, NY, USA, 501–506. https://doi.org/10.1145/3017680.3017779

[31] H. L. III. Roediger and J. Karpicke. 2006. Test-enhanced learning: Taking memory tests improves long-term retention. *Psychological Science* 17 (2006), 249–255. https://doi.org/10.1111/j.1467-9280.2006.01693.x

[32] Karthikeyan Umapathy and Albert D. Ritzhaupt. 2017. A Meta-Analysis of Pair-Programming in Computer Programming Courses: Implications for Educational Practice. *Trans. Comput. Educ.* 17, 4, Article 16 (Aug. 2017), 13 pages. https://doi.org/10.1145/2996201

[33] T. van Gog and L. Kester. 2012. A test of the testing effect: Acquiring problem-solving skills from worked examples. *Cognitive Science* 36 (2012), 1532–1541. https://doi.org/10.1111/cogs.12002

[34] T. van Gog, L. Kester, K. Dirkx, V. Hoogerheide, J. Boerboom, and P. P. J. L. Verkoeijen. 2015. Testing after worked example study does not enhance delayed problem-solving performance compared to restudy. *Educational Psychology Review* 27 (2015), 265–289. https://doi.org/10.1007/s10648-015-9297-3

[35] T. van Gog and J. Sweller. 2015. Not new, but nearly forgotten: The testing effect decreases or even disappears as the complexity of learning materials increases. *Educational Psychology Review* 27 (2015), 247–264. https://doi.org/10.1007/s10648-015-9310-x

[36] David Weintrop and Uri Wilensky. 2015. To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-based Programming. In *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15)*. ACM, New York, NY, USA, 199–208. https://doi.org/10.1145/2771839.2771860

[37] M. West and C. Zilles. 2016. Modeling student scheduling preferences in a computer-based testing facility. In *Proceedings of the Third ACM Conference on Learning at Scale*. 309–312. https://doi.org/10.1145/2876034.2893441

[38] C. L. Wooldrige, J. M. Bugg, M. A. McDaniel, and Y. Liu. 2014. The testing effect with authentic educational materials: A cautionary note. *Journal of Applied Research in Memory and Cognition* 3 (2014), 214–221. https://doi.org/10.1016/j.jarmac.2014.07.001

[39] C. Zilles, R. T. Deloatch, J. Bailey, B. B. Khattar, W. Fagen, C. Heeren, D. Mussulman, and M. West. 2015. Computerized Testing: A Vision and Initial Experiences. In *Proceedings of the American Society for Engineering Education (ASEE) 2015 Annual Conference*. https://doi.org/10.18260/p.23726