Increase Performance and Retention: Teach Students How To Study

Albert Lionelle
Colorado State University
lionelle@colostate.edu

Sudipto Ghosh
Colorado State University
sudipto.ghosh@colostate.edu

Shannon Ourada
Colorado State University
sourada@rams.colostate.edu

Westin Musser
Colorado State University
westin.musser@colostate.edu

ABSTRACT

Intervention in the form of changing one's teaching style is beneficial for boosting student grades and retention. However, in spite of the availability of multiple intervention approaches, a key hindrance is reliance on the belief that students know how to study.

We dedicated time and resources to not only teach the discipline of Computer Science, but also to teach students how to study using techniques grounded in psychology. We offered a one-credit "booster" course to students taking CS 2: Data Structures. Through direct advisor intervention based on the first exam grade, students were encouraged to take the booster course along with traditional interventions. We then tracked student growth across exams for the course as students were learning and being held accountable to study techniques not often emphasized in Computer Science.

The students continued to increase their grades throughout the semester relative to the students who chose to not take the booster class. The students who were targeted for intervention but did not take the booster course continued to have lower grades throughout the semester, and only 41% of them passed the course. Students who participated in the booster course showed a 31% rate of growth across the semester, taking a failing grade to a passing grade, with 100% passing the course with a C or above. These results show a significant influence to help students succeed, which led to higher retention and increased grades. If we want students to truly succeed, we must teach them to study.

CCS CONCEPTS

• **Social and professional topics** → *Computational thinking*; *Computer science education.*

KEYWORDS

Computing education, CS2, Intervention, Psychology, Study Techniques

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGCSE 2022, March 3–5, 2022, Providence, RI, USA
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9070-5/22/03…$15.00
https://doi.org/10.1145/3478431.3499340

ACM Reference Format:
Albert Lionelle, Sudipto Ghosh, Shannon Ourada, and Westin Musser. 2022. Increase Performance and Retention: Teach Students How To Study. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2022), March 3–5, 2022, Providence, RI, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3478431.3499340

1 INTRODUCTION

Research has documented that success in Computer Science (CS) programs is determined in part by previous achievement, such as those shown between CS 1 and CS 2. However, there are additional predictors of student success and retention within university environments such as self-efficacy, motivation, grade goals, academic-related skills, perceived social support, and effort regulation [24, 25]. Most of these skills are often not directly taught to students, but instead indirectly developed throughout their lives. Geitz et al. addresses that these indirect skills promote mastery style practices similar to a growth mindset [9].

Particularly with CS 2, a major predictor of student success is past CS 1 performance, including a correlation of topics between courses [15]. Given that the performance of CS 1 skills is essential for CS 2 performance, we believe it is important to develop an intervention for students who may still lack skills for success, even if they passed CS 1, as not all students come out of CS 1 equally. A key foundation of good educational practice is providing effective support and feedback to students [34]. The importance of intervention has been recognized with numerous papers proposing and evaluating different types of interventions in Computer Science [13, 21, 32, 33].

A typical intervention, such as a TA-led study group that focuses on reteaching student content or presenting the content in a different manner to students until they understand, falls under a common intervention category in Szabo et al. taxonomy of intervention types [32]. However, given the importance of academic skills, it is our belief that failure stems from a missing foundation component often assumed of college students, which is the knowledge of how to study. Education often focuses on teaching students what to study, not how to study.

Our premise is that in addition to presenting content we must teach students how to learn and study new topics, focusing on general skills needed for collegiate success. If a student is taught skills for how to study and be self-sufficient they will demonstrate greater self-efficacy throughout their college career and life.

In this research, we evaluate three semesters of intervention focusing on teaching students that how they studied was just as important, if not more important, than what they studied. Starting Spring 2020 we developed a *booster* course attached to our CS 2: Data Structures course whose primary focus was to teach students how to learn new content in Computer Science. While there are numerous types of study habits that the course could have focused on teaching, we opted to teach habits encouraged by Roediger et al. [26] and others due to their documented success in STEM fields [2, 3, 11, 23, 28–31, 35], with the Reflection technique already being heavily studied and successful in Computer Science Education [5, 8, 17–20, 22]. These principles and techniques were: Spacing, Interleaving, Practiced Retrieval, Elaboration, and Reflection.

A booster course, sometimes termed department-led study group, has the goal of boosting student grades from failing to passing. Our goal for the study group was to teach students how to learn, while the main CS 2 course focused on teaching them what to learn. Any style of booster course seeks to improve student grades as a measurable outcome. Looking at this outcome, we document the massive success for students who adopted these practices into their typical study habits across three semesters of teaching students how to learn. This new learning-to-learn style course is designed to be repeatable with details on its implementation provided in Section 4. The results strengthen our belief that for student success and retention departments must spend time helping students learn the tools they need to be successful in their college career, and in turn, successful in our Computer Science programs promoting both retention and long-term success.

2 RELATED WORKS

Zingaro et al. [36] propose six learning goals for a basic data structures course based on reviewing CS 2 courses at a variety of institutions. Layman et al. [15] present an analysis of CS 1 and CS 2 grade point statistics. CS 1 grades have a statistical relationship with CS 2 grades. Scores in a modified CS 1 concept inventory were significantly correlated with CS 2 grade points. A second analysis was also performed using feedback from CS 2 students on CS 1 preparation and CS 2 challenges. Similarly, the instructors of our CS 2 booster class relied on students taking an exam focused on CS 1 content to determine the types of intervention needed for CS 2 due to their observed association between CS 1 skills and CS 2 performance.

Szabo et al. [32] present the results of a Systematic Literature Review on interventions in Computer Science classrooms based on 129 papers. They presented a taxonomy of intervention types and identified the two most popular types of interventions. One is based on technical cooperation between courses and the other is focused on how to change the way course content is presented to students. Most teaching assistant led study groups or booster classes fall under changing the way content is presented to students. Our booster class is different in that our goal is teaching students how to study with a reduced focus on reiteration of content.

Cottam et al. [4] investigate the effect of peer tutoring on helping CS 1/2 students build a sense of community, succeed in coursework, and enhance their confidence in taking further courses in their major. With a relatively low cost, peer tutoring can improve retention.

Cukierman and Thompson [6] report on the Academic Enhancement Program at Simon Fraser University through which they introduce students to basic learning theory and strategies during first year CS courses. Students also have the option to take an elective component to learn about a topic of their choice related to student success and wellness. The program was studied over a period of three years using student surveys and academic advisor interviews and was found to benefit students and improve retention.

Guo [10] describes the relationship between student performance in CS 1 and CS 2 courses at the University of Berkeley and factors related to their background, prior grades, and the utilization of interventions. Four types of interventions were used. For selected students in CS 1 and CS 2 Computer Science Mentors (CSM), a student-run organization, provided volunteers to lead small-group discussions. In CS 2, weekly group tutoring sessions were held by course staff members. Weekly one-on-one tutoring sessions were offered first-come-first-served by students from a CS pedagogy course. The fourth type of intervention was a guerilla section, where students worked on worksheet problems in groups. Students could not move on to the next question until everyone in the group understood the answer. None of these interventions were associated with a statistically significant increase in performance. The Mentoring sections were associated with a slight decrease in performance.

Mani and Mazumder [17] incorporated meta-cognition into their undergraduate and graduate level courses by having students report on their confidence levels for assignments and tests, and analyzing the confidence levels. Students were able to manage their time effectively and instructors could choose appropriate strategies to meet desired learning outcomes.

3 LEARNING TO LEARN FOUNDATION

The book *Make It Stick* encourages readers to use psychology to improve retention of material through eight different practices to improve memory: retrieving, spacing, interleaving, elaboration, generation, reflection, calibration, and mnemonic devices [1]. Of these eight, five are heavily repeated in Cognitive Science literature. These are categorized by Roediger and Pye. as spacing, interleaving, practiced retrieval (testing), elaboration, and reflection [27]. The booster course utilized these five techniques as the foundation to teach students how to learn. The following highlights some of the literature surrounding the benefits of these five techniques across multiple fields.

The spacing effect describes the benefit in memory retrieval of topics when those topics are studied with a gap between sessions. The process of forgetting, combined with forced recall of the topic enhances student long term memory of that topic. In contrast to spacing, most students study using massed or blocked practice, conventionally known as "cramming" [31]. The spacing effect is one of the oldest findings in experimental psychology dating back to 1885 with work done by Ebbinghaus [7]. There is an extensive body of evidence on the benefits of using spacing, and Cepeda et al. provide an updated overview of using spacing to enhance recall [2]. However, the benefits between spacing and massed practice are a bit

more nuanced as short term recall is benefited by large block studying. This means students see immediate benefit, even if long term recall is not there. Schmidt and Bjork showed that with progressive testing recall decreased when they studied in blocks as compared to increasing results when students practiced spacing [30]. This leads to the conclusion that massed practice is better for short term memory and spacing for long term memory. Furthermore, it has been noted that spacing of material is one of the most powerful methods people can use to increase long term memory without increasing the total amount of time dedicated to studying [11].

In conjunction with spacing, students are encouraged to interleave topics by including a variation of topics in a study session. For example, a student who studies math by learning multiplication and then studies by learning division as compared to a student who studies both topics at the same time intermixing what they are learning. It has been shown that interleaving of these topics improves recall in multiple domains, including mathematics [28, 29]. It is also believed a strength of interleaving is that students are better at discerning and figuring out the problem, causing fewer discrimination errors [27]. Using the math example, a student who practices the same type of problem with variation already knows the goal of the problem, causing them to skip the step in which they have to discern the goal of the problem. Interleaving problem types means a student must first discern the goal of the problem before working on the problem. In an ideal world students combine both interleaving and spacing for all study sessions by interleaving topics while spacing out their study sessions.

Practiced retrieval, better known as testing, is often misunderstood as a means of evaluation of progress and knowledge in an area. Stated another way, testing by itself is a way to force memory retrieval which then enhances memory. Roediger et al. provides an overview of ten different benefits of testing, only one of which is meant to track student progress. [26]. Furthermore, it is shown that more frequent low-stakes tests are better than the occasional midterm, encouraging long term memory retrieval [1, 14, 16, 27]. Unfortunately, testing is the least intuitive of the practices. Students when surveyed often prefer to "restudy" or review material they have previously studied, such as going over notes, slides, and rereading content. It is believed familiarity of content causes the false illusion of knowledge, and students often assume mastery of content before they actually know it [12, 31].

Elaboration involves asking questions about the topics and seeking answers to those questions. Stated another way, it is the "why" question: why does something work the way it does, and then seeking the answer to this question. It is also about developing a body of information based on course content in which the student is forced to seek answers not readily given in the course. Elaboration has been shown to help with both retention and understanding of material [23].

Reflection or self-explanation requires individuals to monitor and describe either out loud or internally their learning process. Going beyond class content, they are asked to question their own understanding of the content. How much do they really know? What do they know, and more importantly, what do they not know? What are ways they can improve? Similar to elaboration, this strategy requires the student to be an active learner, even when reading book material. Such practice is known to elicit improved recall and understanding [3]. Even more, reflection improves applying knowledge to vastly different problems, termed far-transfer of information between topics, and was studied within mathematics. Wong et al. [35] gave students questions to ask while reading. Those in the questioning group performed better than those in the standard group when applying their knowledge to related domains that were not yet covered in studying [35]. The benefits of reflection after exams and assignments have been extensively studied in CS Education [5, 8, 17–20, 22].

Overall, the premise is that applying these practices is an "inexpensive" change to a student's study techniques that will help them learn and retain memory. However, the largest caveat is that both instructors and students naturally gravitate towards ineffective learning techniques due to the short term benefits [27].

4 HOW TO TEACH STUDENTS HOW TO LEARN

Starting Fall 2019, the instructors of our CS 2: Data Structures course began providing students by week 3 of the course a review exam, which is essentially the final of the CS 1 Introduction to Programming course. While the exam is the same, the questions varied by ample use of question banks. It is called the "review exam" as the first three weeks of the semester does a quick review of the final topics in CS 1, emphasizing Object Oriented Programming. This allows us to not only confirm what students retained from CS 1, but also to determine the types of interventions that would best help the student be successful in our program.

Students who scored below 70% on the exam were recommended by the instructor to go back and retake CS 1. For multiple reasons, most students did not do so. Thus, starting Spring 2020, the authors developed a "booster course" for students to help them perform better in CS 2 instead of asking them to retake the previous course.

The goal of the booster course is to teach students how to learn and, in particular, study for exams in Computer Science. The booster course was designed to teach five principles of recall, apply these principles in how they study for CS 2, and be held accountable to their study practice for CS 2. The ultimate goal of the course is to teach study techniques they could use for any course, though there was an emphasis placed on the CS 2 course's topics in all discussions and examples, giving students additional opportunities to go over course content. It is our belief that anyone can be proficient at Computer Science with the right study habits.

4.1 Weekly Meetings

Twice a week for one hour, a Graduate Teaching Assistant (GTA) meets with students and presents one of the five principles using relevant research papers (the same ones presented in Section 3), and slides built for the course. Then using the principle as a foundation, they introduce students to an assignment that was meant to teach them the principle, along with helping them study for CS 2. Overall, there are 11-12 weeks of the booster course, giving 22-24 contact points with the GTA and students. The GTA cycles back to each principle going more in depth on the benefits with every pass.

4.2 Assignment 1: Problem Solving Cards

The primary assignment for students is essentially a flashcard assignment. However, the connotation of the flashcard terminology is focused on definitions and memorization. Our goal was not to have students memorize as we consider Computer Science a problem solving based discipline. Given a set of information, can a computer scientist reason their way to an answer?

In order to build problem-solving cards students were asked to spend time after each CS 2 lecture (3 days a week) and come up with 3-5 exam-like questions. These questions could not be definition-based. Instead, students were asked to use coding examples with how input or output would need to be changed, conduct a more in-depth analysis of a data structure, or include anything else that came to mind. If they needed to understand a concept, they were asked to use it in context and avoid creating a question that required defining the concept. For five nights a week, students were asked to "shuffle" their deck of cards and practice studying for 20 minutes.

This single assignment addresses four of the five principles.

Spacing: Studying 20 minutes a night across five nights is the practice of spacing study habits. They were told to not remove cards from their expansive decks in effort to force recall of older topics.

Interleaving: The shuffling action is meant to promote interleaving, encouraging the mixing of topics in a randomized manner. Thus, when looking at a card, a student would first have to discern the problem being asked.

Elaboration: Thinking like the instructor can be a challenging task, but one that forces the student to ask the "why" of the topic.

Practiced Recall: At the heart of every self-quizzing activity, like flashcard style assignments, is practiced recall. Students had to force their minds to recall information instead of going over their notes.

The biggest challenge was to help students develop their cards from memorization-based to problem solving focused questions. The GTA would spend ample time teaching students how to think more like instructors in designing questions and also providing sample cards, which were then used in class, giving the GTA the opportunity to demonstrate how they would work through the problem aspect of the question. Most students would add to their deck, and students were encouraged to trade examples and cards with each other expanding their own decks. By the end of the semester, students could share their own cards easier.

While the image is a bunch of study cards in a deck of cards, it proved difficult to capture code on a typical 3x3 or 4x4 style card. Most students would adapt programs online to help them, though a single program did not fit their needs. In either case, we did not require any specific format for the cards and just focused on the content they were creating.

4.3 Assignment 2: Journal

For their second major assignment, students were asked to keep a journal of their study habits. The goal of the reflective journal was to help the students practice **reflection**, so they knew we wanted to hold them accountable for their study habits. They wrote down the time they spent studying every night including any insights. Once a week, they were asked to write a paragraph or more reflecting on what they had learned throughout the week, and topics they needed to work on. The TA would go through the reflections making sure they were studying and comment on their reflections if anything stood out to them. Their grade for the course was simply completing the journal including reflections.

5 EMPIRICAL EVALUATION

In order to evaluate the the booster class, we analyzed student performance across three semesters: Spring 2020, Fall 2020, and Spring 2021. Across the three semesters, a total of 674 students took CS 2, mostly in an online or hybrid environment due to COVID 19. The results from the first exam in CS 2 were used to invite students to self select into the following groups.

- Standard: Students who scored over 70% on the review exam. While they were made aware the booster course was happening and open, they were not explicitly invited. 467 students fell into the standard group. This group is included for comparison, but the primary comparison groups are between Recommended and Booster.
- Recommended: Students who scored below 70% on the review exam were encouraged to take the booster course. 176 students opted to not take the booster course. This group is our baseline group.
- Booster: Students who chose to take the booster course and their initial review exam grade was below 70%. 31 students opted to take the booster course and completed the booster course. This is our intervention group. Any student who opted to take the course, but did not show up or participate at all were not included in the data set.

Throughout the semester student progress was tracked. For Spring 2020 there were four exams in the CS 2 course. Fall 2020 and Spring 2021 aligned to have seven exams: one for each major topic covered and the final exam. For the three semester analysis, we look at the combined Review Exam and Final Exam scores. In Fall 2020 and Spring 2021, we examined the progress across all the exams to determine if there was a distinguishing point between topics and student performance.

The primary research questions we seek to answer are:

(1) Does participation in the booster class increase student performance throughout the course, even when new topics are introduced to all groups?
(2) Does participation in the booster class reduce the drop out rate of students in the course?

5.1 Grades and Retention

The Booster group was the only group that did not have course drops. The Standard group had 2% drop out. The Recommended group had the highest drop out rate at 32%.

Table 1 shows the entire grade break down across the groups. Furthermore, in order for students to continue onto CS 3 and any other course that requires CS 2, our department requires students to have a C or better in the course. As such, we define passing as a C grade or better in CS 2, and anything under a C grade (D, F, D/W) is not passing the course. Table 2 details the percentages of each group based on overall A,B,C, and Not passing.

Defining passing grades as a grade with a C or above, over half of the students (59%) who should have taken the booster class but

Table 1: CS 2 Grades For All Students in Spring 2020, Fall 2020, and Spring 2021.

	BOOSTER	RECOMMENDED	STANDARD
	N=31	N=176	N=467
A	2	5	163
B	15	33	163
C	15	36	80
D	0	16	16
F/WD	0	33/55	34/11

didn't, failed to pass the CS 2 course. This is noticeably different from both the Standard group where 13% did not pass, and the Booster group where 100% passed. Using a Chi-Squares test to compare the grades, it is a significant difference. The p-value is < .00001, and the result is significant at $p < .05$.

5.2 Exam Analysis

The only exams that were common between Spring 2020, Fall 2020, and Spring 2021 are the review exam and the final exam. For both they were held mostly consistent with minor updates to the question banks for the exams. However, in Fall 2020 the course moved to topic-based exams, increasing the 4 previous exams to seven total exams, each covering a different topic. The topic-based approach also allowed the instructors to have a stronger understanding of how the students were doing on topics while increasing the number of exams that factored into student grades. At the same time, each exam was worth less to the overall student grade, giving students more leeway to do poorly on their review exam. The following analysis does not take exam weighting into account as the goal is to treat all the exams as equally important.

Figure 1 shows the results of the review exam to the final exam across all three semesters. 58% was the median grade for the students who chose to take the booster class, and 53% was the median grade for the students who chose to not take the booster class. The students who were not recommended to take the booster had 82% as their median grade. When comparing the grades of the Booster group to the Recommended group using a T-test, the result was not significant with a p-value of .28. This shows the Booster group and the Recommended group started in the same place, and there wasn't a significant difference between the groups looking at their grades.

For the Standard group, the Median grade went up on the final exam by 1% increasing to 83%. Essentially, there is very little change if students start out passing the course. The Recommended group of students dropped to 46% for their median grade on the final exam, well below passing. The students who took the booster course had

Table 2: Percentage of Passing and Not-passing Grades

	BOOSTER	RECOMMENDED	STANDARD
A	6%	2%	35%
B	45%	19%	35%
C	48%	20%	17%
D,F,D/W	0%	59%	13%

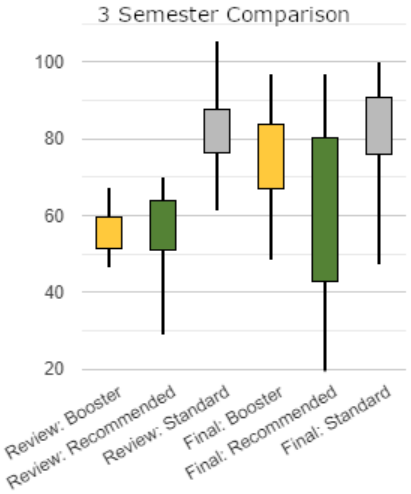

Figure 1: Review and Final Exam comparisons across 3 semesters

a median grade of 73% on the final exam. When running a T-test between the exam grades for the Booster Group students and the Recommended Group of students, it was found that the p-value is .02, and the result is significant at $p < .05$. This demonstrates that the Booster group and Recommended group that started out equivalent at the beginning of the semester grew to show a significant difference between the students' grades and scores by the end of the semester.

When looking at growth as the percent change between the review exam to the final exam, the Booster student group has a 31% growth, well above any other group. After removing all zeros from the Recommended group, there was about a 13% change for the students, and the Standard group remained fixed with a 0% change across their group. Across all three semesters, students who opted to take the booster course were able to turn their failing grades

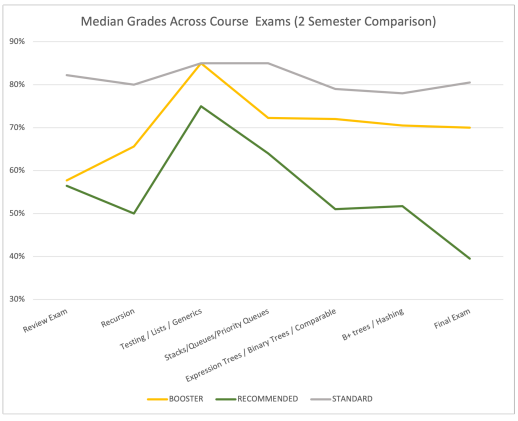

Figure 2: Median Exam Grades Across Topics Fall 2020 and Spring 2021

Table 3: Topic-wise Median Grade Across Exams Fall 2020 and Spring 2021

	BOOSTER	RECOM.	STAND.
Review Exam	58%	56%	82%
Recursion	66%	50%	80%
Testing / Lists / Generics	85%	75%	85%
Stacks / Queues / Priority Queues	72%	64%	85%
Expression Trees / Binary Trees / Comparable	72%	51%	79%
B+ trees / Hashing	71%	52%	78%
Final Exam	70%	40%	81%
Average Across Exams	**71%**	**58%**	**82%**

into passing grades, leading to the question, is that improvement immediate or is there consistent growth throughout the semester?

Looking at Fall 2020 and Spring 2021, it is possible to break down the student growth by topic as each exam focused on a specific topic taught in the course. The exams were not designed to be cumulative except for the final exam. Figure 2 shows the median grade for each group across the exams for the semester.

The exams are as follows: Review is an emphasis on Object Oriented Programming including inheritance, abstraction, and encapsulation. The other topics (recursion, testing (JUnit)/lists/generics, stacks/queues/priority queues, expression trees/binary trees/comparable, B+ trees/hashing) are exactly as stated. The final is comprehensive. Table 3 shows the median grades on the exams.

A common factor we have seen in our courses is that Recursion is often difficult until students take our Discrete Structures course, a course that comes after CS 2 in our curriculum. This result is reflected in the drop for both the Standard students and the Recommended students. At the same time, the Booster class students went up when taking the recursion exam. All groups went up for testing/lists/generics, which may be a reflection of a recent emphasis on testing and lists in CS 1.

Moving past lists on to what is considered newer topics, and the foundation of CS 2, both the Standard and Booster groups drop slightly. The Recommended group starts to drop considerably with every new topic introduced. The Recommended group had 63% earn below a C on the final exam. Additionally, except for the testing exam, the Recommended group never had a median grade over 70%.

Looking at the topic exams overall, the Booster Group was able to bring up the median exam grade and keep it there throughout the semester. The students who were Recommended to take the booster course, but chose not to take it show a continued decrease, and the students who did not receive directed recommendations to take the booster class started out fine, struggled a bit, and then ended fine. They also continued to outperform the students who did need the interventions the booster course provided.

6 THREATS TO VALIDITY

For the Spring 21 cohort about half the students were already introduced to the techniques of the booster course. Starting Fall 2020, we began introducing the learning techniques presented in the booster course to a subset of students taking CS 1. We found of those in the Spring 21 CS 2 cohort that scored under 70 points on the review exam 64% were students who did not have previous experience with the learning techniques. While a previous background with the techniques may have been helpful for students, we did not distinguish when making recommendations. Furthermore, looking across multiple semesters helps reduces inconsistencies and this external threat to validity. Overall, the noticeable difference on the review exam warrants further exploration as these techniques taught in a different setting still show student improvement in the following course.

While these techniques work to encourage students to study problem solving and work on recall, a common question is if this also improves programming ability. This study does not address programming ability, as to do so would introduce a construct validity threat. The way CS 2: Data Structures is setup at our university, students are given multiple opportunities to turn in their coding assignments and labs. Both attendance and performance are graded, though due to multiple submissions, performance is more a matter of a deadline. Our results focus on exam performance as the booster course techniques are designed to help with exams and problem solving. While there is an argument that a good exam tests programming ability, the direct influence on programming ability would be an open area to explore more in the future.

7 CONCLUSION

It is sometimes said that "a good student is a good student." Meaning some students come in with a strong foundation and will always perform to the best of their ability while others do not. We believe what makes a good student is a solid foundation of study skills and habits and those skills can be taught. For three semesters we invited students who performed poorly during their first exam in CS 2 to join a booster class that focuses on boosting their study skills and habits. Students would still attend CS 2 as normal and they would attend a booster class session twice a week that kept students accountable to certain study habits and techniques.

The results were significant. Across three semesters, all students who participated fully in the booster course, practicing what was taught, did not drop out of CS 2. Students who chose not to participate after they were recommended to take it had a 32% dropout rate. Furthermore, looking at passing as C or above, all students across the three semesters who participated in the booster class passed CS 2 with a C or above, while those who choose not to participate only had a 41% passing rate.

Performance wise, students who participated in the booster class went from a 58% median grade on their first exam to a 73% median grade on the final exam, showing a 31% rate of growth across the semester. The students who did not take the booster class scored a 46% median grade on the cumulative final exam.

In this experience, we found that taking the time to teach students how to study, while also holding them accountable for those habits makes a significant difference in their success. If we want students to succeed, especially students who may not have been given the opportunity to fully develop good study habits before joining our programs, this study shows promising results on why departments should invest in our students' ability to study to be successful during their college experience.

ACKNOWLEDGMENTS

This project was partially funded by a Best Practice Grant from The Center for Inclusive Computing at Northeastern University, and we would like to thank the Center for their support and guidance. Any opinions, findings and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the Center or partner institutions.

IRB Protocol Number: 20-10285H

REFERENCES

[1] Peter C Brown, Henry L Roediger III, and Mark A McDaniel. *Make it stick*. Harvard University Press, 2014.

[2] Nicholas J Cepeda, Harold Pashler, Edward Vul, John T Wixted, and Doug Rohrer. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological bulletin*, 132(3):354, 2006.

[3] Michelene TH Chi, Nicholas De Leeuw, Mei-Hung Chiu, and Christian LaVancher. Eliciting self-explanations improves understanding. *Cognitive science*, 18(3):439–477, 1994.

[4] Joseph A Cottam, Suzanne Menzel, and Janet Greenblatt. Tutoring for retention. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, pages 213–218, 2011.

[5] Michelle Craig, Diane Horton, Daniel Zingaro, and Danny Heap. Introducing and evaluating exam wrappers in cs2. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, page 285–290, New York, NY, USA, 2016. Association for Computing Machinery.

[6] Diana Cukierman and Donna McGee Thompson. The academic enhancement program: encouraging students to learn about learning as part of their computing science courses. In *Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education*, pages 171–175, 2009.

[7] Hermann Ebbinghaus. Memory: A contribution to experimental psychology. *Annals of neurosciences*, 20(4):155, 2013.

[8] Katrina Falkner, Rebecca Vivian, and Nickolas J.G. Falkner. Identifying computer science self-regulated learning strategies. In *Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education*, ITiCSE '14, page 291–296, New York, NY, USA, 2014. Association for Computing Machinery.

[9] Gerry Geitz, Desirée Joosten Ten Brinke, and Paul A. Kirschner. Changing learning behaviour: Self-efficacy and goal orientation in PBL groups in higher education. *International Journal of Educational Research*, 75:146–158, jan 2016.

[10] Allen Guo. Analysis of factors and interventions relating to student performance in CS1 and CS2. Master's thesis, EECS Department, University of California, Berkeley, May 2020.

[11] Nate Kornell. Optimising learning using flashcards: Spacing is more effective than cramming. *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition*, 23(9):1297–1317, 2009.

[12] Nate Kornell and Robert A Bjork. The promise and perils of self-regulated study. *Psychonomic Bulletin & Review*, 14(2):219–224, 2007.

[13] Wanda M Kunkle and Robert B Allen. The impact of different teaching approaches and languages on student learning of introductory programming concepts. *ACM Transactions on Computing Education (TOCE)*, 16(1):1–26, 2016.

[14] Douglas P Larsen, Andrew C Butler, and Henry L Roediger III. Repeated testing improves long-term retention relative to repeated study: a randomised controlled trial. *Medical education*, 43(12):1174–1181, 2009.

[15] Lucas Layman, Yang Song, and Curry Guinn. Toward predicting success and failure in cs2: A mixed-method analysis. In *Proceedings of the 2020 ACM Southeast Conference*, pages 218–225, 2020.

[16] Frank C Leeming. The exam-a-day procedure improves performance in psychology classes. *Teaching of Psychology*, 29(3):210–212, 2002.

[17] Murali Mani and Quamrul Mazumder. Incorporating metacognition into learning. In *Proceeding of the 44th ACM technical symposium on Computer science education*, pages 53–58, 2013.

[18] Joshua Martin, Stephen H Edwards, and Clfford A Shaffer. The effects of procrastination interventions on programming project success. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*, pages 3–11, 2015.

[19] Laurie Murphy and Josh Tenenberg. Do computer science students know what they know? a calibration study of data structure knowledge. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITiCSE '05, page 148–152, New York, NY, USA, 2005. Association for Computing Machinery.

[20] Jennifer Parham, Leo Gugerty, and D. E. Stevenson. Empirical evidence for the existence and uses of metacognition in computer science problem solving. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, SIGCSE '10, page 416–420, New York, NY, USA, 2010. Association for Computing Machinery.

[21] Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson. A survey of literature on the teaching of introductory programming. In *Working group reports on ITiCSE on Innovation and technology in computer science education*, pages 204–223. 2007.

[22] James Prather, Brett A. Becker, Michelle Craig, Paul Denny, Dastyni Loksa, and Lauren Margulieux. What do we think we think we are doing? metacognition and self-regulation in programming. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*, ICER '20, pages 2–13, New York, NY, USA, 2020. Association for Computing Machinery.

[23] Michael Pressley, Mark A McDaniel, James E Turnure, Eileen Wood, and Maheen Ahmad. Generation and precision of elaboration: Effects on intentional and incidental learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13(2):291, 1987.

[24] Michelle Richardson, Charles Abraham, and Rod Bond. Psychological correlates of university students' academic performance: A systematic review and meta-analysis. *Psychological Bulletin*, 138(2):353–387, 2012.

[25] Steven B. Robbins, Huy Le, Daniel Davis, Kristy Lauver, Ronelle Langley, and Aaron Carlstrom. Do Psychosocial and Study Skill Factors Predict College Outcomes? A Meta-Analysis. *Psychological Bulletin*, 130(2):261–288, 2004.

[26] Henry L Roediger III, Adam L Putnam, and Megan A Smith. Ten benefits of testing and their applications to educational practice. In *Psychology of learning and motivation*, volume 55, pages 1–36. Elsevier, 2011.

[27] Henry L Roediger III and Mary A Pyc. Inexpensive techniques to improve education: Applying cognitive psychology to enhance educational practice. *Journal of Applied Research in Memory and Cognition*, 1(4):242–248, 2012.

[28] Doug Rohrer, Robert F Dedrick, Marissa K Hartwig, and Chi-Ngai Cheung. A randomized controlled trial of interleaved mathematics practice. *Journal of Educational Psychology*, 112(1):40, 2020.

[29] Doug Rohrer and Kelli Taylor. The shuffling of mathematics problems improves learning. *Instructional Science*, 35(6):481–498, 2007.

[30] Richard A Schmidt and Robert A Bjork. New conceptualizations of practice: Common principles in three paradigms suggest new concepts for training. *Psychological science*, 3(4):207–218, 1992.

[31] Bennett L Schwartz, Lisa K Son, Nate Kornell, and Bridgid Finn. Four principles of memory improvement: A guide to improving learning efficiency. *IJCPS-International Journal of Creativity and Problem Solving*, 21(1):7, 2011.

[32] Claudia Szabo, Nickolas Falkner, Antti Knutas, and Mohsen Dorodchi. Understanding the effects of lecturer intervention on computer science student behaviour. In *proceedings of the 2017 ITiCSE conference on working group reports*, pages 105–124, 2018.

[33] Arto Vihavainen, Jonne Airaksinen, and Christopher Watson. A systematic review of approaches for teaching introductory programming and their influence on success. In *Proceedings of the tenth annual conference on International computing education research*, pages 19–26, 2014.

[34] Henry M. Walker. Acm retention committee retention of students in introductory computing courses: Curricular issues and approaches. *ACM Inroads*, 8(4):14–16, October 2017.

[35] Regina MF Wong, Michael J Lawson, and John Keeves. The effects of self-explanation training on students' problem solving in high-school mathematics. *Learning and Instruction*, 12(2):233–262, 2002.

[36] Daniel Zingaro, Cynthia Taylor, Leo Porter, Michael Clancy, Cynthia Lee, Soohyun Nam Liao, and Kevin C Webb. Identifying student difficulties with basic data structures. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*, pages 169–177, 2018.