Investigating the Impact of Employing Multiple Interventions in a CS1 Course

Sophia Krause-Levy, Leo Porter, Beth Simon, and Christine Alvarado
University of California, San Diego

ABSTRACT

Given the long-standing concern about students failing introductory programming courses, there is a need for interventions that may aid those students. In this work, we examine the potential benefit of three interventions based on prior computing education research (CER) or STEM education research literature: mindset interventions, the use of "Thinkathons" as an alternative to programming labs, and metacognitive interventions to encourage more productive study habits. We conducted an in-class study that controlled for both time-on-task and selection bias to investigate the potential benefits of integrating these interventions into the existing footprint of an introductory computing course. Despite the previously reported promise of the interventions we implemented, our findings were that in this context these techniques had only a mild positive effect for some students. We discuss possible reasons why these techniques are less successful than instructors might hope and argue for the need for more research on this topic.

CCS CONCEPTS

• **Social and professional topics → Computing Education**.

KEYWORDS

CS1, intervention, thinkathon, metacognitive

ACM Reference Format:
Sophia Krause-Levy, Leo Porter, Beth Simon, and Christine Alvarado. 2020. Investigating the Impact of Employing Multiple Interventions in a CS1 Course. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11–14, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3328778.3366866

1 INTRODUCTION

The problem of student retention and success in early programming courses is a challenging and long-studied problem. Decades of work has uncovered barriers students face in CS1 courses, including conceptual [39], pedagogical [30], and social [13] barriers. Since then, a myriad of improvements have been shown to help improve retention and student success [14, 28, 31, 32].

However the problem is far from solved. CS1 failure rates remain high [43] and participation from students from some groups remains low [27]. Struggling students face challenges on a number of fronts including social-psychological (specifically, inclusion), metacognitive, and cognitive (deep conceptual understanding).

Students with less prior computing experience and women generally have a lower sense of belonging in the field (inclusion) [4, 18], while women also report a lower self-efficacy in computing (metacognition) [4]. Women and students from racial and ethnic groups underrepresented in computing often face a hostile culture and are more likely to internalize any struggle as a sign that they do not belong in the discipline [13].

On the cognitive side, solid understanding of fundamental programming concepts in early programming courses is important for students' later success [21, 41]. Unfortunately, with enough time and assistance, many lower-performing students succeed on programming assignments (PAs) with a "hack until the program works" approach, and come away without understanding the underlying concepts [23]. This can hurt them as they try to build on these concepts, and in their performance on written exams, where they cannot rely on the compiler or autograder.

In the last decade both mindset and cognitive interventions have shown some promise in addressing inclusion and cognitive challenges both in and outside of computing. First, growth mindset and values interventions have been shown to improve student performance and retention in STEM courses, particularly for students from underrepresented groups [2, 3, 8, 12, 16, 17]. On the other hand, a recent large multi-institutional study found that growth mindset interventions improved interest in CS but not performance [5]. Second, the concept of a "Thinkathon"—a session in which students solve progressively more difficult problems on paper—was shown to have a positive impact on students' exam grades, and students perceived a high amount of value from it [7]. However, this initial experience report was not a formal study and suffered from threats to validity including selection bias and time-on-task.

This study seeks to explore the following research question: Can combining previously identified promising intervention techniques within a standard structure of a CS1 course significantly increase students' performance and what value, if any, do students perceive from these interventions?

We implemented a quasi-controlled experiment by intervening in the weekly 50-minute closed-lab session of an introductory programming course ("CS1") at a large research university in North America. We replaced on-computer coding exercises with paper-based scaffolded code comprehension and writing exercises (following the Thinkathon model) targeting the same learning outcomes. We also included metacognitive tasks to support study skill development and inclusion and values reflections to foster growth mindset and belonging. In particular, we used values interventions and study skills interventions. We compared this model to a model of the course that used solely coding-based labs without any reflection. Perhaps surprisingly, we found no strong evidence that these

SIGCSE '20, March 11–14, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6793-6/20/03…$15.00
https://doi.org/10.1145/3328778.3366866

interventions generally improved students' performance, though we found no evidence that they hurt. In addition, students generally viewed the intervention labs less favorably than the standard labs.

This work has three main contributions. First, we performed a careful implementation and study of a combination of promising interventions in a CS1 course. The results conflict (at least some) prior findings, which allows us to gain a more nuanced understanding about when these interventions may be successful and when they may not. Second, as we found students did not appreciate the interventions, we offer possible explanations for why and what might be done differently in the future. Third, our findings suggest that these interventions might have a positive effect for women, providing another reason to continue exploring this line of work.

2 BACKGROUND

2.1 Interventions

Designing effective interventions for struggling students is notoriously challenging. One reason is that intervention strategies that have shown high potential for success require significant resources and commitment from both the educator and student. One promising intervention for low-performing students is Supplemental Instruction (SI), a peer-facilitator led program where students receive additional coaching on how to approach challenging material. SI participation is widely shown to correlate with higher exam grades and lower failure rates [9]. Yet SI requires significant support from the teaching team as well as effort and time from the students.

Part of the benefit of SI is that it focuses not only on content, but also on helping students improve their study skills. Lighter-weight interventions that use targeted and well-timed study skills feedback have also shown to be effective in some cases. One study showed that students who met with professors after their midterm exams to discuss changes to their study strategies improved their subsequent midterm score significantly relative to students who only received an email about study skills and those who received no intervention [10]. Another study found that psychology students who chose to attend an in-person study skills training course midway through the term improved their subsequent exam grades [6].

Recently, a line of lightweight interventions based on psychology have gained attention for their potential to increase students' interest, and potentially performance, in their courses. Growth mindset interventions are one intervention technique specifically aimed at helping students develop a mindset that intelligence is malleable, rather than fixed, and to see failure as an opportunity for growth rather than a sign of not being good enough. Early work suggested that growth mindset based interventions might improve academic outcomes [11], but more recent work has failed to show this [38].

Recent studies have also shown the promise of lightweight mindset interventions that target students' values, particularly for first-generation and underrepresented minority students in STEM classes. In two studies, a simple values affirmation intervention in which students wrote about values that were personally important to them reduced the achievement gap for first-generation students and underrepresented minority students in an introductory biology class [16, 19]. A similar intervention based on Expectancy-Value Theory [44] in which students wrote about why the course material was personally relevant to them showed similar results [17]. Other work has explored the value of providing metacognitive scaffolding to aid in programming assignment completion [25, 34].

The success of lightweight metacognitive and mindset interventions in computer science (CS) is not clear. Exam wrappers—a post-exam reflective technique designed to improve students' study skills—showed no effect on students' performance in introductory CS courses in a carefully designed two-university study [40]. Meanwhile, growth mindset interventions have had mixed results in CS. Cutts et al. found students' test scores improved with a six week mindset intervention [8]. In contrast, a correlation study examining the relationship between students' course grades and their mindsets in 3rd year CS courses found a non-significant relationship between mindset and success [20]. Another intervention conducted across multiple institutions also found no significant difference in the mindsets between experimental and control groups [37]. Notably, a recent, large, multi-institutional study on growth mindset interventions in C courses found that these interventions increased students' interest in CS but did not improve their performance [5].

Given these conflicting results and the varying study designs, it is important to continue to explore the contexts in which these interventions may have value. It is also worth exploring whether null results could be because each intervention on its own was too small to produce a result, and whether combining multiple interventions might give sufficient power to yield a change.

2.2 Code Writing and Code Understanding

The impact of understanding early concepts on later performance has been demonstrated in several studies. The theory of learning edge momentum says that students who do better early on tend to continue to do better and widen the gap from those who struggle early on because the concepts tend to build on each other [35]. This phenomenon has been revealed in analyses of final exam questions, where early concepts appear on the majority of the questions on the exam, even those designed to focus on the concepts learned later in the course [29, 33].

Student who struggle in CS1 exhibit patterns of just trying to get the code to work, rather than deeply understanding why [23]. In contrast, students who succeed tend to have mastery goals for learning the course content [45], and likely exhibit behaviors associated with gaining a deeper understanding of the code behavior.

Studies that have explored the relationship between code tracing, code explaining, and code writing [22, 24, 26, 36, 42] have shown that code tracing is necessary for students to be able to write code successfully. As such, Thinkathon interventions aim to help students with deepening their understanding of code behavior through code reading, tracing, and explaining exercises. Our study explores the context of engaging students in code understanding for the sake of deep concept learning as part of weekly labs. This stands in contrast to the original Thinkathon intervention which did so in the context of helping students prepare for the final exam [7].

3 INTERVENTION

With the knowledge that many interventions in CS1 fail, we designed a multi-faceted intervention to combat the three previously identified challenges that students face—inclusion, metacognitive, and cognitive—drawing from the most recent knowledge on why

CS1 students struggle and the most promising intervention techniques from CS and other disciplines. Our goal in combining these interventions was to maximize the potential to improve student outcomes, but we discuss the risk of overloading students in Section 6.2. In this section we describe the components of our complete intervention, which was implemented in the closed lab of a 10-week introductory computing (CS1) course. Details of the experiment we ran to evaluate the intervention are given in the next section. Throughout this section we sometimes paraphrase the intervention prompts (without losing their meaning) for brevity; for replication purposes we provide the complete prompts at [1].

3.1 Inclusion

Students responded to two prompts drawn from the literature on growth mindset and expectancy-value theory.

Overcoming Struggle: In week 1, we asked students to respond to a prompt based on the Growth Mindset lesson plan from `https://mindsetkit.org/`: *Write a letter to a future student of your class about a struggle you have overcome. Tell this student your story and give them advice on what they should do next time they encounter an obstacle when learning something new.*

Value Reflection: In week 4, we implemented a utility-value intervention similar to [17] in which students responded to the following prompt: *Explain how what you are learning in [CS1] is important to you, personally. You might (but do not have to) address any of the following: How does it connect to what you are learning in other courses? How is it relevant to your life outside of [CS1]? What will learning this material enable you to create that is valuable to you?*

3.2 Metacognitive

Students engaged in two interventions designed to help them focus on the areas of the course they were struggling with most.

Learning Reflection: In Week 6, right after the midterm exam, students responded to the following prompt: *Reflect back on last week's exam. Choose one problem that you got incorrect that you feel reveals a concept or skill that you are not 100% sure about.*

1. *What concept or skill is your selected problem related to?*
2. *What actions did you take to learn/practice this knowledge or skill in preparation for the exam?*
3. *How confident were you in this before the exam? Why?*
4. *Looking back, what specific actions could you have taken to better learn this skill or concept? What specific actions will you take NOW to better learn this skill or concept?*

Study Skills: In Week 9, to help students prepare for the final exam, we prompted them for two concepts about which they were confused using the following prompt ((1)-(4) were repeated):
In this warm up you will be asked to list two concepts or topics that you feel you have learned well, and two concepts or topics you feel you need more information on/practice with.

1. *Give one topic or concept you feel you have learned well?*
2. *What makes you confident about your learning?*
3. *Give one topic or concept you feel you still need more practice with, or give one specific question you have.*
4. *What specific action(s) will you take BEFORE NEXT WEEK'S LAB to help you learn it better or get your question answered? [Examples of acceptable and unacceptable responses also given.]*

In the following week, students filled out a yes/no form to verify whether they had completed their action(s).

3.3 Thinkathon

We adopted Thinkathons in all the labs from Week 2-10. (Week 1 was a warmup on how to use the lab machines for assignments.) Based on the work of Cutts et al. [7], we had students work in pairs on paper worksheets that asked students to identify terminology, describe what specific line (or lines) of code accomplished, trace through the execution of code and predict its output, identify potential bugs, and explain (at a high level) the purpose of particular pieces of code. We give an example of a problem from a Thinkathon lab in Section 4.1; the complete Thinkathon labs and solutions are available at [1].

4 STUDY DESIGN

To understand whether and how our multi-faceted intervention would affect students' experience and performance in an introductory computing course, we integrated all three interventions into an offering of an introductory computing course (CS1) that uses the Media Computation curriculum [15] in Java at a large US research-intensive public university in the spring of 2019. There were two sections of this course (each with an enrollment of 146 at Week 4), each of which met for two 120-minute class ("lecture") sessions each week. Both sections were taught by the same experienced instructor (one of the authors of this paper). Class session mixed lecture and active learning including peer instruction (based on `peerinstruction4cs.org`). Each class section was further divided into three closed lab sections (for a total of six lab sections of approximately 50 students each) that met weekly for 50 minutes each. Labs were led by a graduate student teaching assistant in an instructional computer lab. Students received a grade for attending lab in their assigned section and on an end-of-lab quiz. Students in the course also completed pre-class reading, weekly programming assignments, and took two midterms and a final exam.

4.1 Experimental Setup

To control for time-on-task and to provide a feasible adoption model (not requiring extra engagement on the part of the instructors or students), we implemented the interventions in the existing footprint of the weekly closed lab session in this course. The instructor randomly selected three of the six lab sections (two from one class section, and one from the other) to complete the inclusion, metacognitive and Thinkathon exercises during their lab time (experimental group). All exercises were done on paper or using Google forms, except in three cases (week 1, and half of weeks 8 and 10) where students (also) performed some of the programming exercises from the control section when it was critical to the programming assignment. Thinkathon exercises were done in pairs, while reflections were done individually. Students in the non-intervention (control group) sections of the lab completed the lab exercises from previous offerings of the course, which included small programming exercises, done in pairs. Students in all lab sections took the same (brief) online quiz at the end of lab. After each lab, the control and experimental versions of the lab materials and solutions were e-mailed to the respective students. Table 1 provides an example of

the differences between the experimental and control groups' lab section exercises and the corresponding programming assignment (PA) all students were required to complete.

4.2 Data Collection

At the beginning of the course we collected baseline data on students' incoming average GPA, major, year in school, race/ethnicity and gender. All of this data was collected from the university registrar database. Gender (where students could select either Female or Male) and race/ethnicity (where students had to select the one choice they most identify with) had been self-reported by the students upon applying to the university. After the course, we obtained students' final exam grades, overall final grade, and responses to an end-of-course survey. All data was deidentified by a third-party (not a member of the research team or the instructor) prior to analysis. Our data set excluded students who opted out of the study as per our approved IRB protocol.

4.3 Course Composition

As this offering of CS1 was in spring quarter, the course's overall population was somewhat atypical for CS1. Of the 222 students who received a letter grade (not including W), 98% were non-CS-majors, though 77% were taking the class to fulfill a major or GE requirement. 53% of the students had sophomore standing, and 41% had junior or senior standing.

The specific student demographics and overall numbers of students were reasonably balanced between the experimental and control groups, as shown in Table 2. Underrepresented minorities (URM) include students who identify as Latinx/Chicanx or Black[1]. We classified students as higher or lower performing based on their incoming GPAs. Students whose incoming university GPA fell in the upper half of the class' GPAs were considered higher performing, and students with an incoming university GPA in the lower half of the class were considered lower performing. The average incoming GPA of students in the control group (3.2) was slightly higher and not statistically significantly different from the average incoming GPA of students in the experimental group (3.1) (two-tailed t-tests assuming equal variance, $p = .25$). Average GPAs for any given subgroup (e.g. women, URM, etc.) also followed this same trend between the experimental and control sections, and none were statistically significantly different.

5 RESULTS

5.1 Performance

We first compared performance of students in the control vs. experimental groups on the following metrics: average course grade, average final exam score, average programming assignment score and percentage of students who withdrew from the course after week 4 or earned a D or an F (WDF rate). We then compared WDF rates and final exam scores between experimental and control groups for each of the subgroups given in Table 2 including women, men, URM, non-URM, low-performing and high-performing. In all cases we used $p < 0.05$ as a threshold for statistical significance.

[1]We did not have any students who identified as American Indian/Native American.

Table 1: Example prompts from week 9 PA and Labs

Component	Prompt
PA	Write public getters and setters for each instance variable in the class Participant: getName, getWeight, getHeight, setName, setWeight, setHeight, getBMI
Control Lab	Write the get and set methods for the major variable in class Student (Done on the lab machines)
Thinkathon Lab	Write the code (including the method header) for a getter method and a setter method for our new instance field – major for the class Student (Done on paper)

Table 2: Number of Students by Group

Group	Experimental	Control
Overall	114	108
URM	23	22
Non-URM	91	86
Female	65	60
Male	49	48
Lower Performer	57	54
Higher Performer	57	54

5.1.1 WDF Rates. We first looked for any correlation between experimental vs. control group and WDF rate in the course. For each group, we calculated the proportion of students who withdrew or received a grade of D or F vs. those who received a passing grade (A+ through C-). We compared these proportions using a χ^2 test. While the WDF rates were lower for the experimental condition (10.5%) than for the control condition (14.8%), this difference was not statistically significant ($\chi^2 = .58, p = .45$). We next ran the same test for each of the subgroups in Table 2. Again, each of the rates was slightly lower for the students in the experimental condition than for students in the control condition, but none of these differences were statistically significant.

5.1.2 Final Exam (and other) Scores. We compared the final exam scores for students in the experimental group to those of students in the control group using a two-tailed t-test assuming equal variance. Average scores for each group are given in Table 3. Overall, final exam scores between the experimental group (74.8%) and the control group (73.6%) were not statistically significantly different.

When we examined the various subgroups to see if there might be a differential effect for different groups, we found that most subgroups also did not show any significant differences between the experimental and the control groups (see Table 3). The exception to this trend was gender; women performed on average 7 percentage points (4 actual points) higher on the final in the experimental section than in the control section (a statistically significant difference), while men performed on average 7 percentage points (4 actual points) *lower* in the experimental section than in the control section (though this difference was not statistically significant). Because of the number of statistical tests we ran, we are hesitant to read too much into these results, but this potential difference motivates future study.

Table 3: Final Exam Averages (of 54 possible points), including only students who took the final exam. * indicates a statistically significant difference between the experimental and control groups ($p < 0.05$).

	Experimental		Control	
Group	*N*	Avg	*N*	Avg
Overall	111	40.41	106	39.73
URM	21	36.44	22	36.85
Non-URM	90	41.33	84	40.49
Female*	**63**	**41.62**	**60**	**37.64**
Male	48	38.81	46	42.47
Lower Perform	55	35.05	53	34.18
Higher Perform	56	45.66	53	45.29

We also looked for performance differences in the programming assignments and overall course grade. We found no statistically significant differences in performance between the overall groups or for any of the subgroups.

5.2 Students' Perceptions

To identify students' feelings towards the interventions, we administered a post-term survey in the final lab section of the quarter. The survey included a section where students rated their agreement, on a 5-point scale, with statements about course components. The three statements relevant to the intervention are shown in Figure 1.

Figure 1 also shows the proportion of students in each group who selected each response for each of the three statements. Because the "Strongly Disagree" and "Disagree" options had very few or zero responses, for our statistical analysis we collapsed the responses into three categories: Strongly Agree/Agree ("Agree"), Neutral, and Strongly Disagree/Disagree ("Disagree"). We compared experimental vs. control group responses using χ^2 tests. As seen in Figure 1, students were generally more positive (pink and red colors on the right) in the control group; however, only the differences in responses to the statement "lab helped me solve my PAs" were statistically significant. This difference makes sense—because the labs in the experimental section were done almost solely on paper, it is not surprising that students did not see as much of a direct connection to their programming assignments.

Table 4 shows rates of agreement and disagreement for each of the subgroups for each of the three statements. The traditional (control group) labs were consistently rated more positively than the experimental group labs, though this difference was only statistically significant in a few cases. It seems that the difference was starker for underrepresented minority students, who had a relatively more positive view of the traditional labs and a relatively more negative view of the experimental labs.

5.3 Threats to Validity

The first threat to validity is the composition of the course, which comprised nearly all non-CS-majors and non-first-year students without prior programming experience. The difference between this population and the course's core audience of first-year majors may impact the efficacy of the interventions; students may have been more driven to simply pass the class than to earn an A.

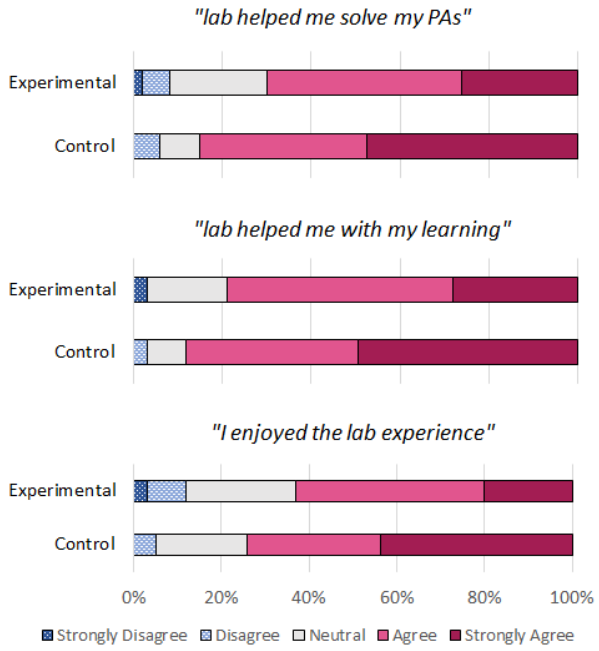

Figure 1: Proportion of each response for each of the three statements about the lab. PA=Programming Assignment

A second threat arises from the implementation of the Thinkathon intervention. Although we had access to the original Thinkathon materials and used them as a guide, our materials differed as they needed to support students learning concepts for the first time—not just an end of the term review. It is also possible that the Thinkathon materials did not accurately match the learning goals of the control labs, introducing an unintended difference between the two groups.

Third, there was likely unintended information flow between the control and experimental groups as students were interleaved in the same active-learning lectures. It is very likely that they shared information about the labs either in class or when studying together.

Finally, the number of statistical tests run makes it difficult to interpret which results might reflect an actual underlying difference. Few of the differences we observed were statistically significant, yet the trends were almost all in the same direction (experimental labs were related to slightly better exam performance and lower WDF rates, but slightly less positively received by students). We must interpret these results as suggestions for further investigation, instead of drawing direct conclusions.

6 DISCUSSION

6.1 Comparison to Original Thinkathon

Our goal was not to replicate the original Thinkathon design, but to explore the value the Thinkathon approach would have if it were not an opt-in, relatively time-unlimited activity. However, we now suspect that some of the key components to the original Thinkathon's success were its link to the exam and the open-ended time frame. The question remains whether the same gains could be achieved for students who did not voluntarily attend these sessions. In any future replication studies, we would at a minimum explicitly draw a link between exam preparation and Thinkathon exercises.

Table 4: Proportion of students who agreed and disagreed with each statement. Bold values indicate a statistically significant difference between the experimental and control sections for a particular statement and subgroup.

Group	*N*		"lab helped me solve my PAs"		"lab helped me with my learning"		"I enjoyed the lab experience"	
	Exp	Ctl	Experimental Disagree/Agree	Control Disagree/Agree	Experimental Disagree/Agree	Control Disagree/Agree	Experimental Disagree/Agree	Control Disagree/Agree
Overall	100	101	**7.0% / 61.4%**	**5.6% / 79.6%**	2.6% / 69.3%	2.8% / 82.4%	10.5% / 55.3%	4.6% / 69.4%
URM	18	22	**0.0% / 56.5%**	**4.6% / 95.5%**	4.4% / 60.9%	4.6% / 90.9%	**8.7% / 34.8%**	**4.6% / 86.4%**
Non-URM	82	79	8.8% / 62.6%	5.8% / 75.6%	2.2% / 71.4%	2.3% / 80.2%	11.0% / 60.4%	4.7% / 65.1%
Female	59	57	4.6% / 66.2%	6.7% / 80.0%	0.0% / 75.4%	3.3% / 83.3%	7.7% / 60.0%	8.3% / 70.0%
Male	41	44	10.2% / 55.1%	4.2% / 79.2%	6.1% / 61.2%	2.1% / 81.3%	**14.3% / 49.0%**	**0.0% / 68.8%**
Lower Perf.	48	47	**5.3% / 56.1%**	**5.6% / 77.8%**	3.5% / 68.4%	1.9% / 77.8%	10.5% / 50.9%	1.9% / 68.5%
Higher Perf.	52	54	8.8% / 66.7%	5.6% / 81.5%	1.8% / 70.2%	3.7% / 87.0%	10.5% / 59.7%	7.4% / 70.4%

We believe that some of the negativity towards Thinkathon labs was due to the expectation that the labs would help students prepare for programming assignments. If we had provided students with a direct connection between the Thinkathon labs and the exams, we might have seen performance improvements and a less negative stance towards Thinkathon labs.

Another key difference is the use of Thinkathon activities during the developmental time of first learning a new computing concept versus stimulating comprehension skills of materials at the end of the term. Perhaps it is beneficial for students to first spend time "just making it work" (as was the case for students in [7]) in order to set them up for deeper understanding activities later.

6.2 Why Didn't Non-Cognitive Activities Help?

We included metacognition and inclusion activities due to the strong evidence in other fields on the positive impacts for underrepresented groups. However, impacts of such interventions specifically in introductory programming courses have been mixed. A large-scale study that was published while we were in the middle of our study also failed to observe performance gains (or gender differences) from growth mindset interventions in introductory computing courses [5]. That paper also found what seems to be common in introductory programming courses: students have a pretty strong growth mindset orientation from the start.

In addition to potentially having a growth mindset from the start, we believe three other factors might have contributed to the limited impact of the interventions. First, the course was already heavily laden with best practices. Peer Instruction, we suspect, already encourages a growth mindset through frequent formative feedback and collaborative learning. Second, the students in the course were mostly non-CS-major students who might not have been as motivated to succeed, and non-first-years who might have developed persistence and self-efficacy through their years in school. Third, it is possible that the interventions were both too numerous and too broadly targeted. Most students did not need all of the interventions we offered, and receiving them all might have overwhelmed them to the point where they could not focus on the intervention that would have helped them.

6.3 Ideas for Future Studies

Replication studies are critical in helping our field understand the robustness of a finding or theory. A single study almost always has threats to validity and is often limited to a particular institution, course, student body, or instructor. Replication studies help us gain confidence and/or better understand nuanced conditions where a finding is applicable. We identify the following considerations to better explore and understand the conditions under which components of our intervention could be more impactful:

- **Replicate in fall term**. First-year students may benefit more from metacognitive supports.
- **Tell students how the activities benefit them**. Both for Thinkathon and non-cognitive activities, explain how these activities are meant to support their success in the course.
- **Use Thinkathon activities as "exam preparation"**. Exam preparation may be better motivation for students.
- **Do not hold Thinkathon in the computer lab**. Students may feel they are losing the opportunity to prepare for their programming assignments by not working on the computer.
- **Design Thinkathons to address all five original goals**. Perhaps support for Mastery Learning, allowing students to work at their own pace, and having students complete the activities individually are necessary conditions for Thinkathon activities to support better test performance.
- **Study the value of lab itself**. The (essentially) null results from this study raise the question of what learning value lab provides. Perhaps students benefit from a social, supportive environment, no matter what they are working on. Or perhaps they would do just as well without it, which would be valuable to know given the resources devoted to it.

7 CONCLUSION

We conducted a well-controlled study of a multi-faceted intervention based on best-practices from the CER and STEM education literature. Despite the seeming promise of the interventions, we found that students generally were not that positive about them and they had only minimal (if any) effect on students' academic outcomes in the course. These results raise questions about next steps in improving outcomes for students in CS1. Further work is needed to fully understand the value and limitations of employing these interventions. Moreover, in this time of high enrollments and large classes, it may be time to step back and reassess our basic assumptions about how to help students learn introductory CS.

8 ACKNOWLEDGEMENTS

We would like to thank the Teaching and Learning Commons for providing the de-identified data for our study and Quintin Cutts for providing the Thinkathon materials used in his study. This work was supported in part by NSF IUSE Grant DUE-1712508.

REFERENCES

[1] 2019. Intervention Materials from this paper. https://researchlinks.page.link/cs1-interventions. (2019). Provided for replication and other use.

[2] Joshua Aronson, Carrie B Fried, and Catherine Good. 2002. Reducing the effects of stereotype threat on African American college students by shaping theories of intelligence. *Journal of Experimental Social Psychology* 38, 2 (2002), 113–125.

[3] Lisa S Blackwell, Kali H Trzesniewski, and Carol Sorich Dweck. 2007. Implicit theories of intelligence predict achievement across an adolescent transition: A longitudinal study and an intervention. *Child development* 78, 1 (2007), 246–263.

[4] Jennifer M. Blaney and Jane G. Stout. 2017. Examining the Relationship Between Introductory Computing Course Experiences, Self-Efficacy, and Belonging Among First-Generation College Women. In *Proceedings of the 48th ACM Technical Symposium on Computer Science Education (SIGCSE '17)*. 69–74.

[5] Jeni Burnette, Crystal Hoyt, V Russell, Barry Lawson, Carol S. Dweck, and Eli Finkel. 2019. A Growth Mind-Set Intervention Improves Interest but Not Academic Performance in the Field of Computer Science. *Social Psychological and Personality Science* (2019), 194855061984163.

[6] Christie L. Cathey, Michelle E Visio, Brooke L. Whisenhunt, Danae L. Hudson, and Carol F. Shoptaugh. 2016. Helping When They Are Listening: A Midterm Study Skills Intervention for Introductory Psychology. *Psychology Learning & Teaching* 15, 3 (2016), 250–267.

[7] Quintin Cutts, Matthew Barr, Mireilla Bikanga Ada, Peter Donaldson, Steve Draper, Jack Parkinson, Jeremy Singer, and Lovisa Sundin. 2019. Experience Report: Thinkathon – Countering an "I Got It Working" Mentality with Pencil-and-Paper Exercises. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. 203–209.

[8] Quintin Cutts, Emily Cutts, Stephen Draper, Patrick O'Donnell, and Peter Saffrey. 2010. Manipulating mindset to positively influence introductory programming performance. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*. 431–435.

[9] Phillip Dawson, Jacques van der Meer, Jane Skalicky, and Kym Cowley. 2014. On the Effectiveness of Supplemental Instruction: A Systematic Review of Supplemental Instruction and Peer-Assisted Study Sessions Literature Between 2001 and 2010. *Review of Educational Research* 84, 4 (2014), 609–639.

[10] Louis Deslauriers, Sara E Harris, Erin Lane, and Carl E Wieman. 2012. Transforming the lowest-performing students: an intervention that worked. *Journal of College Science Teaching* 41 (2012), 80–88.

[11] Carol S. Dweck. 2008. Brainology: Transforming students' motivation to learn. *Independent School* 0, 0 (2008).

[12] Carol S. Dweck. 2008. *Mindset: The new psychology of success*. Random House Digital, Inc.

[13] Kathy Garvin-Doxas and Lecia J. Barker. 2004. Communication in Computer Science Classrooms: Understanding Defensive Climates As a Means of Creating Supportive Behaviors. *Journal on Educational Resources in Computing* 4, 1 (2004).

[14] Mark Guzdial. 2013. Exploring Hypotheses About Media Computation. In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*. 19–26.

[15] Mark Guzdial and Barbara Ericson. 2007. *Introduction to computing & programming in Java: a multimedia approach*. Pearson Prentice Hall.

[16] Judith M. Harackiewicz, Elizabeth A. Canning, Yoi Tibbetts, Cynthia J. Giffen, Seth S. Blair, Douglas I. Rouse, and Janet S. Hyde. 2014. Closing the Social Class Achievement Gap for First-Generation Students in Undergraduate Biology. *Journal of Educational Psychology* 106, 2 (2014).

[17] Judith M. Harackiewicz, Elizabeth A. Canning, Yoi Tibbetts, Stacy J. Priniski, and Janet S. Hyde. 2016. Closing Achievement Gaps with a Utility-Value Intervention: Disentangling Race and Social Class. *Journal of Personality and Social Psychology* 111, 5 (2016), 745–765.

[18] Linda J. Sax, Jennifer Blaney, Kathleen Lehman, Sarah Rodriguez, Kari George, and Christina Zavala. 2018. Sense of Belonging in Computing: The Role of Introductory Courses for Women and Underrepresented Minority Students. *Social Sciences* 7 (2018), 122.

[19] Hannah Jordt, Sarah L. Eddy, Riley Brazil, Ignatius Lau, Chelsea Mann, Sara E. Brownell, Katherine King, and Scott Freeman. 2017. Values Affirmation Intervention Reduces Achievement Gap between Underrepresented Minority and White Students in Introductory Biology Classes. *CBE Life Science Education* 16, 3 (2017).

[20] Antti-Juhani Kaijanaho and Ville Tirronen. 2018. Fixed Versus Growth Mindset Does Not Seem to Matter Much: A Prospective Observational Study in Two Late Bachelor Level Computer Science Courses. In *Proceedings of the 14th Conference on International Computing Education Research*. 11–20.

[21] Sophia Krause-Levy, Sander Valstar, William G. Griswold, and Leo Porter. 2020. Exploring the Link Between Prerequisites and Performance in Advanced Data Structures. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*.

[22] Amruth N. Kumar. 2015. Solving Code-tracing Problems and Its Effect on Code-writing Skills Pertaining to Program Semantics. In *Proceedings of the 20th ACM Conference on Innovation and Technology in Computer Science Education*. 314–319.

[23] Soohyun Nam Liao, Sander Valstar, Kevin Thai, Christine Alvarado, Daniel Zingaro, William G. Griswold, and Leo Porter. 2019. Behaviors of Higher and Lower Performing Students in CS1. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. 196–202.

[24] Raymond Lister, Beth Simon, Errol Thompson, Jacqueline L Whalley, and Christine Prasad. 2006. Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. *ACM SIGCSE Bulletin* 38, 3 (2006), 118–122.

[25] Dastyni Loksa. 2017. Explicitly Teaching Metacognitive and Self-Regulation Skills in Computing. In *Proceedings of the 13th ACM Conference on International Computing Education Research*. 289–290.

[26] Mike Lopez, Jacqueline Whalley, Phil Robbins, and Raymond Lister. 2008. Relationships Between Reading, Tracing and Writing Skills in Introductory Programming. In *Proceedings of the Fourth International Workshop on Computing Education Research*. 101–112.

[27] Jane Margolis. 2010. *Stuck in the shallow end: Education, race, and computing*. MIT Press.

[28] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. 2002. The Effects of Pair-programming on Performance in an Introductory Programming Course. In *Proceedings of the 33rd ACM Technical Symposium on Computer Science Education*. 38–42.

[29] Andrew Petersen, Michelle Craig, and Daniel Zingaro. 2011. Reviewing CS1 Exam Question Content. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*. 631–636.

[30] Leo Porter, Cynthia Bailey Lee, and Beth Simon. 2013. Halving Fail Rates Using Peer Instruction: A Study of Four Computer Science Courses. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. 177–182.

[31] Leo Porter, Saturnino Garcia, Hung-Wei Tseng, and Daniel Zingaro. 2013. Evaluating student understanding of core concepts in computer architecture. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*. 279–284.

[32] Leo Porter and Beth Simon. 2013. Retaining Nearly One-third More Majors with a Trio of Instructional Best Practices in CS1. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. 165–170.

[33] Leo Porter and Daniel Zingaro. 2014. Importance of Early Performance in CS1: Two Conflicting Assessment Stories. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*. 295–300.

[34] James Prather, Raymond Pettit, Brett A. Becker, Paul Denny, Dastyni Loksa, Alani Peters, Zachary Albrecht, and Krista Masci. 2019. First Things First: Providing Metacognitive Scaffolding for Interpreting Problem Prompts. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 531–537.

[35] Anthony Robins. 2010. Learning edge momentum: a new account of outcomes in CS1. *Computer Science Education* 20, 1 (2010), 37–71.

[36] Judy Sheard, Angela Carbone, Raymond Lister, Beth Simon, Errol Thompson, and Jacqueline L Whalley. 2008. Going SOLO to assess novice programmers. In *ACM SIGCSE Bulletin*, Vol. 40. 209–213.

[37] Beth Simon, Brian Hanks, Laurie Murphy, Sue Fitzgerald, Renée McCauley, Lynda Thomas, and Carol Zander. 2008. Saying Isn't Necessarily Believing: Influencing Self-theories in Computing. In *Proceedings of the Fourth International Workshop on Computing Education Research*. 173–184.

[38] Victoria F. Sisk, Alexander P. Burgoyne, Jingze Sun, Jennifer L. Butler, and Brooke N. Macnamara. 2018. To What Extent and Under Which Circumstances Are Growth Mind-Sets Important to Academic Achievement? Two Meta-Analyses. *Psychological Science* 29, 4 (2018), 549–571.

[39] E. Soloway. 1986. Learning to Program = Learning to Construct Mechanisms and Explanations. *Commun. ACM* 29, 9 (1986), 850–858.

[40] Ben Stephenson, Michelle Craig, Daniel Zingaro, Diane Horton, Danny Heap, and Elaine Huynh. 2017. Exam Wrappers: Not a Silver Bullet. In *Proceedings of the 48th ACM Technical Symposium on Computer Science Education*. 573–578.

[41] Sander Valstar, William G. Griswold, and Leo Porter. 2019. The Relationship Between Prerequisite Proficiency and Student Performance in an Upper-Division Computing Course. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. 794–800.

[42] Anne Venables, Grace Tan, and Raymond Lister. 2009. A Closer Look at Tracing, Explaining and Code Writing Skills in the Novice Programmer. In *Proceedings of the Fifth International Workshop on Computing Education Research Workshop*. 117–128.

[43] Christopher Watson and Frederick WB Li. 2014. Failure rates in introductory programming revisited. In *Proceedings of the 19th Conference on Innovation and Technology in Computer Science Education*. 39–44.

[44] Allan Wigfield and Jacquelynne S. Eccles. 2000. Expectancy–Value Theory of Achievement Motivation. *Contemporary Educational Psychology* 25, 1 (2000), 68 – 81.

[45] Daniel Zingaro, Michelle Craig, Leo Porter, Brett A. Becker, Yingjun Cao, Phill Conrad, Diana Cukierman, Arto Hellas, Dastyni Loksa, and Neena Thota. 2018. Achievement Goals in CS1: Replication and Extension. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 687–692.