Standards-Aligned Instructional Supports to Promote Computer Science Teachers' Pedagogical Content Knowledge

Satabdi Basu
satabdi.basu@sri.com
SRI International

Daisy Rutstein
daisy.rutstein@sri.com
SRI International

Carol Tate
carol.tate@sri.com
SRI International

Arif Rachmatullah
arif.rachmatullah@sri.com
SRI International

Hui Yang
hui.yang@sri.com
SRI International

ABSTRACT

The rapid expansion of K-12 CS education has made it critical to support CS teachers, many of whom are new to teaching CS, with the necessary resources and training to strengthen their understanding of CS concepts and how to effectively teach CS. CS teachers are often tasked with teaching different curricula using different programming languages in different grades or during different school years, and tend to receive different professional development (PD) for each curriculum they are required to teach. This often leads to a lack of deep understanding of the underlying CS concepts and how different curricula address the same concepts in different ways. Empowering teachers to develop a deep understanding of CS standards, and use formative assessments to recognize common student challenges associated with the standards, will enable teachers to provide more effective CS instruction, irrespective of the curriculum and/or programming language they are tasked with using. This position paper advocates supporting CS teacher professional learning by supplementing existing curriculum-specific teacher PD with standards-aligned PD that focuses on teachers' conceptual understanding of CS standards and ability to adapt instruction based on student understanding of concepts underlying the CS standards. We share concrete examples of how to design standards-aligned educative resources and instructionally supportive tools that promote teachers' understanding of CS standards and common student challenges and develop teachers' formative assessment literacy, all essential components of CS pedagogical content knowledge.

CCS CONCEPTS

• **Social and professional topics → Student assessment**; **K-12 education**; **Computer science education**.

KEYWORDS

CS teacher professional learning, pedagogical content knowledge, formative assessment literacy, CS standards, K-12 CS education, algorithms and programming

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 ACM 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 2–March 5, 2022, Providence, RI, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-8214-4/21/06…$15.00
https://doi.org/10.1145/3430665.3456347

ACM Reference Format:
Satabdi Basu, Daisy Rutstein, Carol Tate, Arif Rachmatullah, and Hui Yang. 2022. Standards-Aligned Instructional Supports to Promote Computer Science Teachers' Pedagogical Content Knowledge. In *The 53rd ACM Technical Symposium on Computer Science Education (SIGCSE 2022), March 2–March 5, 2022, Providence, RI, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3430665.3456347

1 INTRODUCTION

States and school districts across the country are ramping up efforts to increase Computer Science (CS) offerings in K-12 [5]. With this rapidly increasing demand for high quality K-12 CS education, it is critical to provide practicing teachers, many of whom were not originally certified to teach CS, with the tools, resources, and training they need. The introduction of the K-12 CS standards [2] has mapped the domain of K-12 CS, but teachers still need support on understanding several aspects of the standards and how to effectively teach the standards. They need help understanding the fine-grained learning targets associated with the broad CS standards, what it means to cover a standard using different programming representations, common student challenges associated with the standards, and how to assess and scaffold students' progress towards proficiency on the standards. In recent studies, teachers reported some of their biggest problems as having insufficient CS subject matter knowledge, struggles with assessing student work, meeting student needs on an individual basis, inadequate planning time, feelings of isolation, and IT challenges [14, 28].

CS teachers are often required to simultaneously teach different curricula using different programming languages in different grades (e.g., MyCS using Scratch in Grade 6 and CS Discoveries using JavaScript in Grade 7) or to switch between teaching different curricula across different school years. Teachers tend to receive separate professional development (PD) for each CS curriculum they teach, but often lack a deep understanding of the underlying CS concepts and standards aligned with the curricula and how different curricula address the same concepts and standards in different ways. To benefit teachers with varied CS backgrounds using a variety of CS curricula, it is imperative that we supplement existing curriculum-specific in-service teacher PD with standards-aligned PD that focuses on teachers' conceptual understanding of CS standards and common student challenges associated with the standards. Empowering teachers with such standards-aligned PD will help teachers better leverage the curriculum-specific PD, reduce the burden of switching between curricula, and enable teachers to

provide more effective CS instruction, irrespective of the curriculum and/or programming language they are required to use for teaching.

The ability to measure and track students' progress towards target CS standards and address students' challenges is another critical area where CS teachers need support to better meet their learners' needs. An understanding of CS standards and concepts is a necessary but not sufficient component of formative assessment literacy – the ability to assess students' progress and use the assessment information to inform instruction [20]. Research indicates that teachers new to a content area and/or teaching practice often face significant challenges when engaging in robust assessment practices in their instruction [6, 20]. These challenges are relatively well understood in other disciplines, such as math and science, as are PD strategies for helping teachers develop and improve these practices (e.g., [11]), but the same is not true in CS. The CSTA Assessment Task Force's report [27] also calls for the design and implementation of PD opportunities to help develop and improve CS teachers' assessment literacy and use of classroom assessments.

This position paper calls for supporting CS teacher capacity building by supplementing existing curriculum-specific in-service teacher PD with standards-aligned PD that focuses on both teachers' conceptual understanding of CS standards and their ability to adapt instruction based on student understanding of concepts underlying the CS standards. We situate our position on the need for this work in existing literature on CS teacher challenges and what constitutes effective CS instruction as well as findings from a focus group we conducted with middle school CS teachers. We share concrete examples of how to design standards-aligned educative resources and instructionally supportive tools that promote teachers' understanding of CS standards and their formative assessment literacy, both essential components of CS pedagogical content knowledge (PCK). While we call for teacher supports supporting all of the standards, our examples are situated in the Algorithms and Programming strand of the CS standards.

2 RELEVANT RELATED WORK

2.1 Literature Review

Effective CS instruction requires the ability to blend knowledge of content, pedagogy, and technology (e.g., programming environments, data manipulation tools) into an understanding of how particular CS concepts can be represented using relevant technologies, adapted to the diverse interests and abilities of learners, and presented for instruction in an accessible form that addresses common student misunderstandings. This ability is the essence of CS technological pedagogical content knowledge, or TPACK [18], which is particularly challenging for new CS teachers [14, 28]. Possessing a deep understanding of CS standards and demonstrating formative assessment literacy are both important components of CS TPACK and effective CS instruction. Table 1 highlights teacher standards (what CS teachers should know and be able to do) [3] that emphasize the need for CS teachers to understand student-level CS standards and be able to adapt instruction to address student challenges.

Due to a shortage of certified CS teachers, teachers are often recruited from other subject areas and they struggle with CS TPACK due to insufficient CS content and technological knowledge and CS-specific pedagogical strategies [9]. Even existing technology and computer-literacy teachers who are accustomed to leading classes on keyboarding and using office applications are likely to be challenged by lessons on algorithms, data science, and impacts of computing [7] [23]. Many studies have attempted to respond to CS teachers' challenges. For example, PD4CS [21] developed PD for the AP CSP course, Leyzberg and Moretti [17] developed PD targeting CS1 concepts, and TALECS PD [1] targeted the Exploring CS curriculum. However, most studies have involved high school teachers, and have focused primarily on specific curricula and increasing teachers' CS content knowledge. Many PD workshops and online training modules often focus on adoption of specific CS curricula, programming environments, and technology-rich tools at the expense of promoting pedagogical practices such as formative assessment literacy. In a recent survey of a national sample of high school CS teachers, 70% of respondents reported that their PD emphasized "deepening their own CS knowledge, including programming," but fewer than 50% reported "learning about difficulties that students may have with particular CS ideas or practices" and "monitoring student understanding during CS instruction" as areas of emphasis [4].

Table 1: CS Teacher Standards Emphasizing an Understanding of CS Content Standards and Formative Assessment

Teacher Standard 4: Instructional Design	
4a	Analyze CS curricula in terms of CS standards alignment, accuracy, completeness of content, cultural relevance, and accessibility
4b	Design and adapt learning experiences that align to comprehensive K-12 CS standards
4g	Develop multiple forms and modalities of assessment to provide feedback and support. Use resulting data for instructional decision-making and differentiation

Formative assessment literacy comprises the ability to elicit, understand and use assessment data to make decisions about instruction [20]. It is articulated as part of CS teacher standards, but teachers are grossly under-prepared to consistently and effectively use assessment results to inform their planning and instruction. A large survey of secondary teachers [13] concluded that the assessment data that teachers gathered was mostly analyzed at a superficial level, leaving the potential for informing instruction untapped. Research on teachers' use of assessment data to inform instruction found that an absence of teacher PD has hampered teachers' ability to use data effectively [15], drawing attention to the need for PD that promotes assessment literacy for teachers.

2.2 Teacher Focus Group Findings

In addition to what we learned from the literature, we also conducted three virtual focus groups [8], 90-120 minutes each, with a total of 11 middle school teachers from diverse CS backgrounds. The focus groups involved small-group discussions and teachers jotting down thoughts in collaborative documents. The entire sessions were video recorded and analyzed using a constant comparative approach [25] to identify critical features and emerging themes across the eleven teachers.

The focus groups were designed to obtain feedback on teacher needs and what kinds of resources would be most useful to teachers. These conversations confirmed that our hypothesized teacher needs matched teachers' actual experiences. We learned about teachers' unique contexts for teaching CS and the primary challenges they face. Additionally, we discussed teachers' familiarity with and perceptions of CS standards, planning processes for teaching CS, instructional strategies, and assessment practices. These discussions helped inform the design of a set of teacher resources to support teachers in furthering their understanding of CS standards and formative assessment literacy.

Teachers described various challenges, the most common ones being struggling with understanding CS content and how students learn it, feeling unequipped to assess student learning in CS, and difficulties with motivating students.

Challenges with understanding CS content and how students learn it. Teachers' challenges with understanding CS concepts and standards stem from a lack of formal experience in learning and teaching CS. One teacher mentioned that not having a firm grasp of CS concepts makes it difficult for her to unpack the standards and differentiate the curriculum to address the wide range of students' prior experience. Another mentioned being unable to anticipate students' misconceptions about the content.

Challenges with assessing student learning in CS. Teachers' lack of CS knowledge also contributes to difficulty with assessing CS learning and helping students who are stuck. Teachers indicated that they rarely used any form of formal or rigorous assessment, whether summative or formative, to evaluate student learning. Most teachers we spoke with indicated lack of familiarity with assessment practices and difficulty with evaluating student work, particularly programs. One teacher added that lacking understanding of how to determine levels of proficiency of student work adds to the difficulty. Teachers indicated that they relied more on students' demonstrated persistence and completion of tasks for grading rather than focusing on the details of the artifacts that students created. Observing students is another method that teachers used to evaluate student learning and monitor when students needed help. A few teachers mentioned coupling this observation method with some peer assessment to help students communicate and learn from each other. Teachers also reported relying heavily on visual checks and verbal feedback.

Several teachers, especially those who embed CS into other courses, reported that CS is not listed on students' report cards, which discourages them from digging deeper into possible CS assessments and committing the time necessary to understand the CS standards more deeply. They did not seem to see the importance of formative assessment as a tool for adapting instruction and providing feedback to students.

Challenges with motivating students. Another challenge identified by some teachers involved the inability to motivate students who quickly give in to frustration. Motivating students requires the ability to pitch instruction to a level that will present an appropriate challenge. This ability depends on deep understanding of both the subject matter and how students learn it.

Despite the challenges described above, teachers described their commitment to exposing students to CS. They generally agreed that CS is an essential subject for students to learn. As one put it, "it's an equity issue and absolutely vital to have..." This belief fueled their interest in learning more about CS standards and student misconceptions and obtaining more guidance to teach CS effectively. These findings about teacher challenges from our focus groups are consistent with those from previous studies (e.g., [28]) and reinforced our belief that providing teachers with a set of educative resources and tools as described in Sections 3 and 4 can be a means to address novice CS teachers' desire to learn more about CS standards, assessments, and student misconceptions, and thus a lever for improving CS instruction.

II. Learning targets for 2-AP-12:
This standard focuses on students' ability to use and comprehend combined control structures, such as nested loops, nested conditional statements, compound conditionals, nested procedures, conditionals nested within loops, and loops nested within conditionals.

Some of the primary learning targets associated with this standard include:

1. **Nested loops (create and debug)**: Ability to create a nested loop and/or debug a given nested loop to represent a given scenario.
2. **Nested loops (output identification)**: Ability to recognize or identify the output of code that includes a nested loop. This includes the ability to identify how many times various action(s) will repeat and what sequence they will repeat in.
3. **Nested conditionals (create and debug)**: Ability to create nested conditional statements and/or debug given nested conditionals to represent a given narrative description
4. **Nested conditionals (output identification)**: Ability to identify the output of nested conditional statements
5. **Compound conditionals (create and debug)**: Ability to create and/or debug a conditional statement that includes logical (AND, OR, NOT) operators to represent a given narrative description
6. **Compound conditionals (output identification)**: Ability to identify the output of a conditional statement that uses logical (AND, OR, NOT) operators as part of the condition
7. **Repeated conditionals (create and debug)**: Ability to create and/or debug a conditional statement that is evaluated repeatedly (conditional inside a loop, or a repeat-until construct)
8. **Repeated conditionals (output identification)**: Ability to identify the output of a conditional statement that is evaluated repeatedly (conditional inside a loop, or a repeat-until construct)
9. **Procedures inside a control structure (create and debug)**: Ability to create and/or debug a procedure that is called within a different control structure (e.g., procedure inside a loop, procedure invoked by another procedure)
10. **Procedures inside a control structure (output identification)**:: Ability to identify the output of a procedure that is called within a different control structure (e.g., procedure inside a loop, procedure invoked by another procedure)

Figure 1: Decomposition of the 2-AP-12 Standard Into 10 Fine-Grained Learning Targets

3 DESIGNING TEACHER EDUCATIVE RESOURCES FOR CS STANDARDS

In order to design teacher-facing educative resources for any CS standard, irrespective of its content area or grade-band alignment, the following represent some essential components to include and considerations to be mindful of.

(1) **Scope of the standard** – Teachers will have different CS backgrounds and varying degrees of awareness about the CS standards; hence it is helpful to start with a description of what content area (e.g., data and analysis, algorithms and programming) and grade band the standard is targeting, and domain-specific vocabulary that teachers should be aware of. Another useful component of the scope is knowing the boundaries of the standard – what students should know from previous grades, what the standard covers, and what

Examples of nested loops learning targets

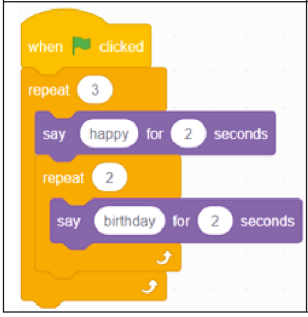
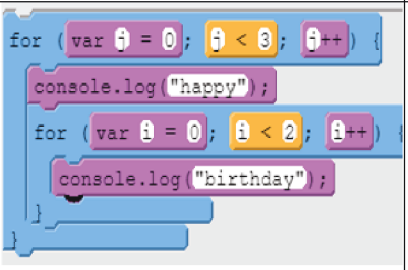
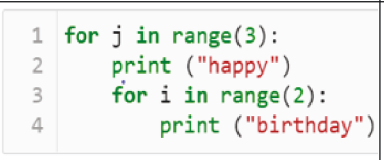
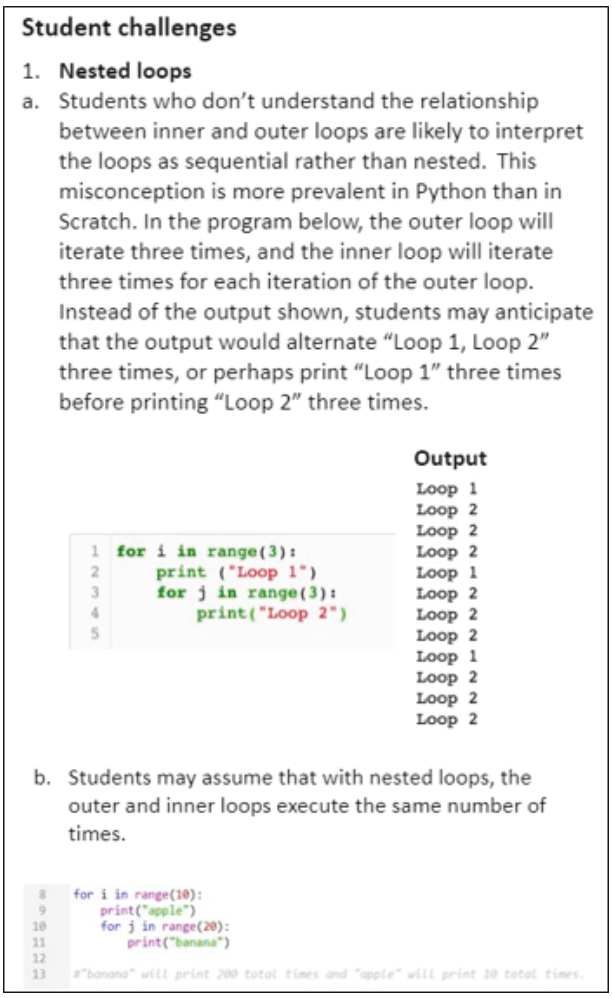
Programming Language: Scratch	Programming Language: JavaScript (CS Discoveries)	Programming Language: Python	Output
			happy birthday birthday happy birthday birthday happy birthday birthday

Figure 2: Examples of the nested loop concept in three different programming languages



Student challenges

1. **Nested loops**

a. Students who don't understand the relationship between inner and outer loops are likely to interpret the loops as sequential rather than nested. This misconception is more prevalent in Python than in Scratch. In the program below, the outer loop will iterate three times, and the inner loop will iterate three times for each iteration of the outer loop. Instead of the output shown, students may anticipate that the output would alternate "Loop 1, Loop 2" three times, or perhaps print "Loop 1" three times before printing "Loop 2" three times.

```
for i in range(3):
    print ("Loop 1")
    for j in range(3):
        print("Loop 2")
```

Output

```
Loop 1
Loop 2
Loop 2
Loop 2
Loop 1
Loop 2
Loop 2
Loop 2
Loop 1
Loop 2
Loop 2
Loop 2
```

b. Students may assume that with nested loops, the outer and inner loops execute the same number of times.

```
for i in range(10):
    print("apple")
    for j in range(20):
        print("banana")

#"banana" will print 200 total times and "apple" will print 10 total times.
```

Figure 3: An example of common student challenges associated with the concept of nested loops

will be covered in future grades. While standards are defined for elementary grades, many students are exposed to CS for the first time in middle school or even later. In such scenarios, it is important for teachers to understand what students should have already learned and help them build their CS foundation before addressing grade level standards.

(2) **Fine-grained learning targets** – Each standard represents a broad performance statement that can be decomposed into a set of granular learning targets representing what students with proficiency on the standard should know and be able to do. Figure 1 shows how one middle school standard from the 'Algorithms and Programming' strand (2-AP-12) can be decomposed into 10 different learning targets encompassing different CS concepts and practices. Often, curricular activities claim alignment with a standard, but only align with a few aspects of the standard. This might result in teachers mistaking what the standard really entails. When vastly different curricular activities claim alignment with the same standard without clarifying what part(s) of the standard they are aligned to, it can be particularly confusing for novice CS teachers with an incomplete understanding of the standard.

(3) **Learning targets in action** – Providing examples of the learning targets in action in different scenarios is critical to teachers developing a deep conceptual understanding of the learning targets. For programming-focused standards, in order to make the resources useful for teachers using different programming languages and to highlight how different languages represent the same concepts, it is critical to provide the examples of learning targets in a few different languages. For example, 'nested loops' is a concept covered by the 2-AP-12 standard and the fact that nested loops refer to one loop nested within another does not change across languages. But, the types and names of loops used in different languages and the syntax of the loops can vary. Figure2 shows examples of nested loops in three different languages (Scratch, JavaScript and Python) that use different representations and syntax but generate the same output.

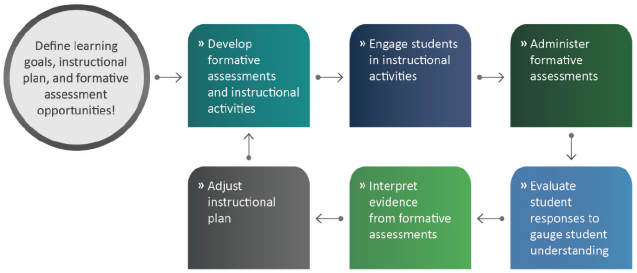


Figure 4: Formative assessment cycle

(4) **Student Challenges** – There is a growing body of research on student misconceptions and challenges in K-12 CS [10, 16, 19, 22, 24]. Deep understanding of a CS standard and effective instruction targeting the standard require that teachers are aware of student challenges known to exist with relation to the standard (challenges have not yet been documented for all CS standards). Awareness of the challenges is a first step towards being able to recognize and address the challenges. Teacher educative resources about CS standards should not only outline related student challenges but also include examples of what the challenges look like in practice (Figure 3) so that teachers can recognize them within student work.

4 DESIGNING FORMATIVE ASSESSMENT LITERACY TOOLS

In the words of Dylan Wiliam [26], "While there are many possible ways in which we could seek to develop the practice of serving teachers, attention to minute-by-minute and day-to-day formative assessment is likely to have the biggest impact on student outcomes" (p. 27). Use of formative assessment is a cyclic process (Figure 4) that involves the identification of important aspects of student learning, the creation (or selection) of assessment tasks to elicit information about student understanding of these aspects, and the use of the resulting information to inform instructional choices, for example, to revisit a certain topic or to address a specific misconception [12]. Content knowledge is a necessary but not sufficient component of formative assessment literacy. One of the key differences between formative and summative assessment lies in the purpose of the assessment. In a summative assessment, the goal is often to provide a score or grade for a student that reflects their understanding. However, the goals of formative assessment are to support student learning, provide students with feedback, and inform instruction.

Designing tools to support CS teachers' formative assessment literacy involves supporting each step of this process including (1) choosing or creating appropriate assessments to use in class, (2) accurately scoring student work and using scored student work to identify student challenges, and (3) deciding what follow-up instructional moves are required to address student challenges. Below, we describe the design of each of these supports.

(1) **Developing or selecting formative assessment tasks** – Some teachers may choose to develop their own formative assessment tasks while others will select from existing available tasks. All teachers will need to ensure that the tasks they use are aligned with the learning target or targets for which they want to measure their students' understanding. As described in Section 3, standards represent broad performance statements. So, it is important that teachers identify which fine-grained learning targets they want to assess and then develop or select tasks aligned with those targets as opposed to broad standards. We suggest a two pronged approach of equipping teachers with tools to help them learn how to develop their own tasks as well as assessment task exemplars to demonstrate what assessments might look like and how evidence of student proficiency might be elicited.

(2) **Assessment design specifications.** To support teachers using different CS curricula that rely on different technological tools and/or programming representations, an assessment design specification for each standard can serve as a template for developing assessment tasks aligned to the standard using any tool or programming representation. They are designed to be adaptable for different formats and types of formative assessment tasks. Assessment design specifications for a standard provide guidance on defining the desired evidence for each learning target aligned with the standard, known challenges to gather evidence about, task features needed to elicit the desired evidence, and ways to vary the difficulty and context of the tasks. The desired evidence for a learning target highlights what we want to see in student performance to determine how well students have attained the target. Task features are the characteristics that must be present in a task to ensure it is able to elicit this desired evidence. For example, if the learning target is about debugging, one task feature is that students must be presented with code that has an error. The task features can serve as a checklist when choosing or developing a task. Variable features, on the other hand, are features that can be varied to measure the learning target in different ways and highlight decisions that must be considered when developing or selecting a task. They can be tied to the context of the class, such as what programming representation is used in the task, or they can be used to modify the format of the task (such as a multiple-choice item versus an open-ended task) or vary the complexity of the code that is presented in the task.

(3) **Assessment task exemplars.** As an example of what assessment tasks derived from the assessment design specifications might look like, it is helpful to provide teachers a set of exemplar tasks for each standard using specific programming representations. To support teachers' knowledge of assessment design, it is important to not only provide the tasks but also connect the design specification to the tasks and describe the design decisions that shaped the tasks. These decisions include learning target(s) to focus on, challenges to target and incorporate into answer choices for tasks, and decisions made for each of the variable features.
Exemplar tasks can demonstrate to teachers how formative assessment tasks can diagnose student challenges, and can provide teachers with ideas for task scenarios and formats (selected responses, program output predictions, program completions, reordering programming instructions, drawing flowcharts, writing pseudocode, constructing written

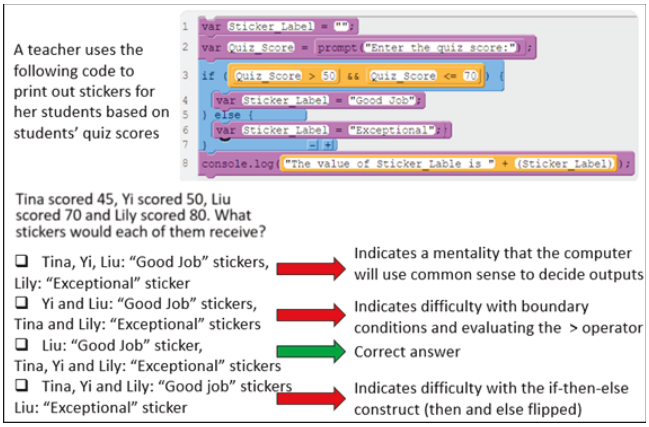


Figure 5: Educative scoring guides

explanations), in particular ones that provide rich and timely information. The assessment design specifications for each standard, coupled with the exemplar tasks, aim to empower diverse teachers to author their own formative assessments, a task many CS teachers find daunting [28].

(4) **Interpreting student work and diagnosing student challenges** – To leverage the full potential of classroom-based assessments, teachers should be able to efficiently score student responses and correctly interpret trends in student scores. Providing teachers with easy-to-use scoring guides that highlight student challenges is a necessity for implementation of formative assessment practices. This helps teachers with an important component of formative assessments — using student scores to identify where students need more support. Scoring guides for promoting formative assessment literacy need to be more educative than simple rubrics that merely assign scores; they should provide teachers with examples of what different levels of proficiency look like for constructed response tasks or programming tasks, and help teachers understand what misconceptions incorrect responses might indicate (e.g., Figure 5).

(5) **Follow-up instructional strategies** - Formative assessment practices are not complete until teachers use their interpretation of student scores to inform instruction. Recommending useful follow-up instructional moves to teachers and modeling what the moves might look like in the classroom can be immensely helpful for all teachers, in particular new CS teachers. Some effective follow-up strategies include (i) having students trace through a program line by line, (ii) having students enact different characters or even different variables in a program, (iii) having students predict program output before running the program, (iv) sharing an incorrect program and having students identify problems, (v) explaining a program line by line by showing how modifications affect the program output, and (vi) facilitating unplugged activities aligned to the standards. The follow-up strategies may not be task-specific but should come with recommended uses for specific types of student errors. The suggested instructional strategies will provide teachers with high-level pointers for how to follow-up with the entire class or with specific groups of students based on students' performance on the assessment tasks.

Together, these resources serve as important instructionally supportive tools for promoting CS formative assessment literacy.

5 DISCUSSION AND CONCLUSION

In the context of a rapidly expanding K-12 CS education movement, this position paper proposes supplementing existing curriculum-specific, content-focused CS PD opportunities with standards-aligned instructional supports that will promote teachers' CS PCK. The paper provides design guidance and examples for developing standards-aligned instructional supports that deepen teachers' general understanding of CS concepts and standards, curricular alignment with standards, common student challenges, and formative assessment practices. Supporting teachers to develop a deeper understanding of CS standards and ability to diagnose and address common student challenges associated with the standards will enable them to provide more effective and engaging CS instruction, irrespective of the curriculum and/or programming language they are tasked with using. This, in turn, will help teachers develop proficiency towards the CS teacher standards that specify what CS teachers should know and be able to do.

Ideally, when teachers develop formative assessment skills, it should bring about a shift in teacher mindsets where evaluating student work goes from being an exercise in assigning grades to that of an useful activity generating information to guide instruction. Formative assessment literacy can help teachers shift their focus from evaluating student work as correct versus incorrect to diagnosing what student challenges may have resulted in an incorrect solution or an incorrectly functioning computational artifact.

While the work described in this paper is an important step towards promoting CS teacher PCK, we recognize that it focuses on a narrow band of what CS teachers need to know and be able to do to provide effective CS instruction. For example, our work does not include critical CS PD components such as promoting CS content knowledge or equitable and inclusive instructional practices. We assume that teachers are already receiving curriculum-specific PD that introduces them to the required content knowledge and advocate supplementing that PD with our standards-aligned PD. Further, we have chosen a specific strand of CS content knowledge related to algorithms and programming because we anticipate it to be an area of particular need for novice CS teachers. It will be important for future work to explore what kinds of resources and instructional supports will benefit other areas of CS content knowledge and other aspects of CS PCK.

Alongside designing these standards-aligned instructional supports for teachers, it is also important to design sustained PD opportunities to help teachers derive the full benefit of the developed resources and tools. Research studies examining the effects of such instructional supports on CS teachers' PCK, self-efficacy, and classroom practices are also needed, as are validated instruments to measure teachers' PCK. Building teacher PCK is an important step toward ensuring equitable access to high quality and engaging CS instruction for all students.

ACKNOWLEDGMENTS

We gratefully acknowledge the middle school teachers who participated in this study. We would also like to thank Dr. Steven McGee and Katya Winkler for supporting the teacher focus groups and recruiting teachers for the focus groups. This material is based upon work supported by the National Science Foundation under Grant No. DRL-2010591. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Nonye Alozie and Jennifer Knudsen. 2020. Middle School Teachers' Perspectives and Experiences on Developing Formative Assessments During Professional Development on a Computer Science Curriculum. In *Society for Information Technology & Teacher Education International Conference*. Association for the Advancement of Computing in Education (AACE), Waynesville, NC, 1077–1083.

[2] Computer Science Teachers Association. 2017. CSTA K-12 computer science standards. Revised 2017. http://www.csteachers.org/standards

[3] Computer Science Teachers Association. 2020. Standards for computer science teachers. https://csteachers.org/teacherstandards

[4] Eric R Banilower, P Sean Smith, Kristen A Malzahn, Courtney L Plumley, Evelyn M Gordon, and Meredith L Hayes. 2018. Report of the 2018 NSSME+. , 442 pages.

[5] Code.org Advocacy Coalition. 2019. Code.org 2019 Annual Report. https://code.org/files/Code.org-Annual-Report-2019.pdf

[6] Christopher DeLuca and Don A Klinger. 2010. Assessment literacy development: Identifying gaps in teacher candidates' learning. *Assessment in Education: Principles, Policy & Practice* 17, 4 (2010), 419–438.

[7] Barry Fishman, Chris Dede, and Barbara Means. 2016. Teaching and technology: New tools for new times. , 1269–1334 pages.

[8] PY Frasier, L Slatt, V Kowlowitz, DO Kollisch, and M Mintzer. 1997. Focus groups: a useful tool for curriculum evaluation. *Family Medicine* 29, 7 (1997), 500–507.

[9] Judith Gal-Ezer and Chris Stephenson. 2010. Computer science teacher preparation is critical. *ACM Inroads* 1, 1 (2010), 61–66.

[10] Shuchi Grover and Satabdi Basu. 2017. Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education*. ACM, Seattle, WA, 267–272.

[11] Margaret Heritage, Jinok Kim, Terry Vendlinski, and Joan Herman. 2009. From evidence to action: A seamless process in formative assessment? *Educational measurement: issues and practice* 28, 3 (2009), 24–31.

[12] Joan Herman. 2013. Formative assessment for next generation science standards: A proposed model. In *Invitational research symposium on science assessment*. ETS, Princeton, NJ, 27 pages.

[13] Nancy R Hoover and Lisa M Abrams. 2013. Teachers' instructional use of summative student assessment data. *Applied Measurement in Education* 26, 3 (2013), 219–231.

[14] Aleata Hubbard. 2018. Pedagogical content knowledge in computing education: A review of the research literature. *Computer Science Education* 28, 2 (2018), 117–135.

[15] Lea Hubbard and Amanda Datnow. 2015. Teachers' use of assessment data to inform instruction: Lessons from the past and prospects for the future. *Teachers College Record* 117, 4 (2015), 1–26.

[16] Tobias Kohn. 2017. Variable evaluation: An exploration of novice programmers' understanding and common misconceptions. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, Seattle, WA, 345–350.

[17] Dan Leyzberg and Christopher Moretti. 2017. Teaching CS to CS teachers: Addressing the need for advanced content in K-12 professional development. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education*. ACM, Seattle, WA, 369–374.

[18] Punya Mishra and Matthew J Koehler. 2006. Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers college record* 108, 6 (2006), 1017–1054.

[19] Monika Mladenović, Ivica Boljat, and Žana Žanko. 2018. Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education and Information Technologies* 23, 4 (2018), 1483–1500.

[20] W James Popham. 2009. Assessment literacy for teachers: Faddish or fundamental? *Theory into practice* 48, 1 (2009), 4–11.

[21] Yizhou Qian, Susanne Hambrusch, Aman Yadav, and Sarah Gretter. 2018. Who needs what: Recommendations for designing effective online professional development for computer science teachers. *Journal of Research on Technology in Education* 50, 2 (2018), 164–181.

[22] Yizhou Qian, Susanne Hambrusch, Aman Yadav, Sarah Gretter, and Yue Li. 2020. Teachers' perceptions of student misconceptions in introductory programming. *Journal of Educational Computing Research* 58, 2 (2020), 364–397.

[23] Alexander Repenning, David C Webb, Kyu Han Koh, Hilarie Nickerson, Susan B Miller, Catharine Brand, Ian Her Many Horses, Ashok Basawapatna, Fred Gluck, Ryan Grover, et al. 2015. Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education (TOCE)* 15, 2 (2015), 1–31.

[24] Juha Sorva. 2018. Misconceptions and the beginner programmer. *Computer science education: Perspectives on teaching and learning in school* 171 (2018), 12 pages.

[25] Anselm Strauss and Juliet Corbin. 1998. *Basics of qualitative research techniques*. Sage Publications, Thousand Oaks, CA.

[26] Dylan Wiliam. 2017. *Embedded formative assessment*. Solution Tree Press, Bloomington, IN.

[27] Aman Yadav, David Burkhart, Daniel Moix, Eric Snow, Padmaja Bandaru, and Lissa Clayborn. 2015. Sowing the seeds: A landscape study on assessment in secondary computer science education.

[28] Aman Yadav, Sarah Gretter, Susanne Hambrusch, and Phil Sands. 2016. Expanding computer science education in schools: understanding teacher experiences and challenges. *Computer Science Education* 26, 4 (2016), 235–254.