Using Programming to Express Mathematical Ideas

As computer-science-for-all initiatives continue to grow (Bers, 2018; Code.org, n.d.; CSforAll, n.d.), many schools are looking for ways to introduce computer science skills and thinking to elementary-age children (ECEP, n.d.; NYCDOE, n.d.). Some initiatives have focused on coding as its own endeavor, not integrated with other subjects like mathematics, science or literacy (Angeli et al., 2016). Increasingly, developers and researchers are exploring ways that programming can be integrated into core subjects (Bers, Govind, & Relkin 2022), though challenges remain to ensure they are mutually supportive of both subjects (Fofang et al., 2020; Sherwood et al., 2021). Our team of teachers and researchers is investigating one such approach that integrates programming and elementary mathematics, developing microworlds for grades 2–5 that treat programming as a language to help children express and investigate mathematical ideas.

In this article, we describe one of the programming environments we have developed, *Number Line* (see Figures 1 and 2) and our observations of second graders' experiences and mathematical thinking while using this tool. We'll discuss our approach to integrating programming into elementary mathematics. Through programming activities in the blocks-based language Snap*!*, children can build their identities as doers of mathematics by constructing their own mathematical ideas, testing them as the computer enacts them, and sharing and discussing those ideas with others. In this microworld, that approach supports second-graders in developing their skills in the domain of whole number concepts and operations, specifically building their fluency adding and subtracting within 20 by creating, testing, and revising programming scripts to solve mathematical puzzles (see supplementary file).

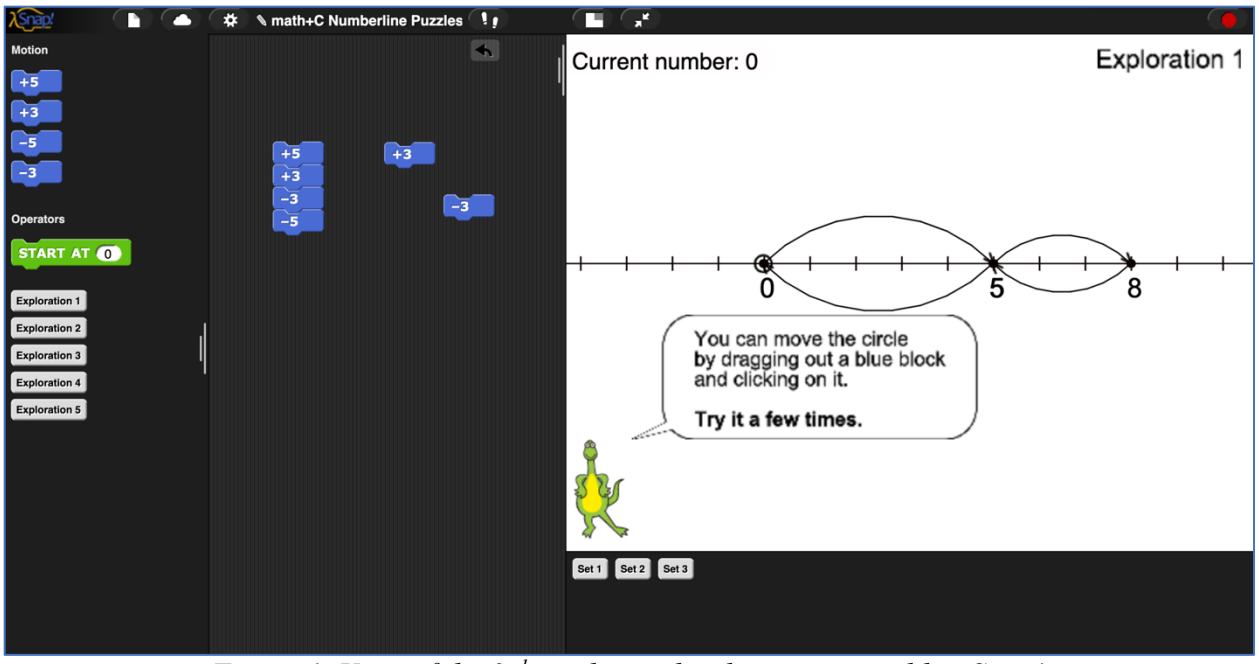

Figure 1. View of the 2nd grade number line microworld in Snap!

Figure 2: Sample puzzles

Integrating programming into elementary mathematics. A programming language is a formal language for expressing certain kinds of ideas—algorithm and abstraction in particular—in a precise way. Learning to program can augment children's expressive power in mathematics and make key ideas accessible. Programming gives children a language with which to express their thinking precisely and have that thinking enacted by the computer, making it easier for them to assess, refine, discuss, and extend that thinking (Cuoco & Goldenberg, 1996; Goldenberg & Carter, 2021).

Further, embedding programming into core mathematics instruction ensures that all children gain experience with programming in developmentally appropriate ways, and increases

access to mathematical ideas by offering different affordances than pencil-and-paper activities. Jottings on paper must be reinterpreted by the child and reenacted in the child's mind; jottings in the form of computer code can be reenacted by the machine. When a child writes "to make an even number you times by 2" on paper, the notation just sits there. It records your idea but doesn't help you perform the next calculation, and doesn't help you make your expression more precise. A program does that and more: expressing our thinking in a program lets us examine it, discuss it with others, revise/debug it, and *refine* it (Goldenberg et al., 2021). In effect, as explained in Cullen, Hertel, and Nickels' framework (2020) for the roles of technology for mathematics education, this use of technology can be characterized as the computer "serving as a tutee", in which children use programming to teach the computer what to do, talking to it in a language it understands (Taylor, 1980).

What is a microworld? A *microworld* provides only the programming tools needed for a specific context. The number line microworld limits the programming blocks that children can access within Snap*!*; children have all, but only, the blocks they need. This limits cognitive distraction and provides a low threshold for entry into programming, while still giving children an authentic programming experience. The microworld also provides three sets of highly mathematical puzzles, increasing in sophistication, for children to solve. Children build *scripts* in order to solve these puzzles, which when run, show the mathematical action on a *stage*, allowing the child to evaluate the results of their choices. In this way, the microworld acts as a "math action technology" (Dick and Hollenbrands 2011; McCulloch et al. 2021), enabling children to explore mathematical concepts and relationships through action on mathematical objects.

The microworld is designed to be easily accessible for teachers, as well as children – teachers need not be expert in programming to use these resources. Because the microworld

limits the blocks children have access to, the interface is easy to learn. The accompanying teacher guide offers guidance on a multi-day sequence of instruction to be integrated with the core mathematics curriculum.

Our microworlds build on decades of exploration of the role programming can play in children's learning. Seymour Papert, the mathematician who co-founded the MIT Artificial Intelligence Laboratory, co-created the Logo programming language in 1967. Logo was designed as a tool for learning – learning mathematics, language, music, art, robotics, science, computer science – and allowed learners to develop projects of all kinds. It was designed to be *low threshold* and *no ceiling* and accessible to novices (Logo Foundation, n.d.). As technology evolved, the idea of programming by assembling blocks led to the development of Scratch, an interpretation of Logo. Scratch – and later ScratchJr and Snap*!* – make programming even more accessible to young children (Resnick et al., 2009). Our microworlds use Snap*!*; those familiar with Scratch will find many similarities between the languages, which bear a family resemblance.

Supporting the development of mathematical identity and agency. This microworld and others we are developing aim to build children's interest and curiosity in both mathematics and computer science – we want all children to see themselves as capable and competent mathematicians, to experience the beauty and joy of creating scripts, to persist in solving challenging problems, and to develop confidence and interest in both domains.

A key design principle for *Number Line* has been to center the locus of control and authority in the child, supporting all children in seeing themselves as capable doers of mathematics – contributing to their development of positive identities as young mathematicians (NCTM, 2020). Learning mathematics involves acquiring knowledge, but also developing

effective ways of thinking, including the ability and inclination to judge whether one's methods and results make sense. Learning to program is similar; even young programmers must be able to judge for themselves whether code does what it is meant to do. *Number Line* is not "gamified" – there are no levels, points or stars; the computer does not evaluate children's solutions. Children experiment, see results, evaluate for themselves whether their scripts have the desired outcome, and debug to get results they want. Authority resides in the child.

We believe the microworlds are tools teachers can use to support children to develop positive mathematical identities and build their sense of mathematical agency (NCTM, 2018). The microwords provide carefully sequenced problem sets that are centered on critical mathematical content for each grade, promote reasoning and problem solving, and provide opportunities for productive struggle with mathematical ideas. These curricular tools – and the engaging programming context they are embedded in – support a child-centered approach, freeing teachers to act as facilitators who circulate among children to observe mathematical thinking in action, provide scaffolding and support for challenge, plan classroom discussions, and consider how to sequence future instruction. Teachers who have used the microworlds report that the experience allows them to see some children engaging differently than they do in paper-and pencil activities, and that they learn more about what their students know as a result. The scripts that children build in Snap*!* make their mathematical thinking visible to others, and enable children to easily share problem-solving strategies and compare the results of different approaches, potentially contributing to their sense of agency in mathematical discussions with peers, with teachers, and in whole-group settings.

The microworlds also promote children's positive mathematics identities by including children's home language in the mathematics classroom. Our development work was in

partnership with linguistically diverse classrooms, with over half the children having a first language other than English. This allowed us to develop and test supports for multilingual learners so that they could engage fully in exploration of the mathematical ideas. All microworlds are available in English and Spanish (some are also available in Portuguese, Ukrainian, and German, with additional languages still in development). Children can program in any available language, and even switch between languages. Children can also have the puzzles read aloud, further facilitating access.

***Number Line* microworld**. This microworld—focused on addition and subtraction on the number line within 20—presents puzzles that require children to navigate the number line. It displays a *palette* of blocks, a *scripting area*, and a *stage* showing a number line with a creature who suggests puzzles (see Figure 1). The ticks on the number line mark consecutive integers; only 0 is labeled to begin.

Four blocks of **+5**, **+3**, **–3**, and **–5** let children move the circle on the number line. Each move draws the proper arc and labels the new number. A **start at** block lets children specify a starting place. Puzzle blocks offer new explorations, puzzles, and challenges (see Figure 2 for some examples). Each puzzle set corresponds to one or two class periods of instruction.

Why ±3 and ±5 blocks? Many of the puzzles ask children to start at one number and move to another—for example, moving from 2 to 9. If we provided a **+7** block, the problem would be trivial. Instead, we limit the blocks to **±3** and **±5**, requiring children to find combinations that work. Even if they start out "just jumping around" – as some do – they remain attentive to where their experiments land them, and then look for opportunities to make more deliberate moves. The *numbers* studied in second grade are often much larger than 3 and 5, but the cognitive challenge of seeking combinations is greater—with many additions and

subtractions playing out in the child's mind in a single puzzle—making it pedagogically sensible to keep the numbers small. And when children run their scripts, small jumps on the number line are easy to see and comprehend quickly.

The choice of **3** and **5** works well in this microworld because they are two small, non-consecutive integers, with no common factor other than 1. Offering **±1** would make the solutions trivial. Offering two consecutive numbers makes it too easy to "get" a 1. Using two numbers that have *any* factor (other than 1) in common – for example, **±6** and **±9** – makes it impossible to land on all integers. The choice of adding and subtracting the *same* number (instead of just offering **+3** and **–5**, which would be sufficient to get all integers) lets children learn to recognize and see power in inverses.

Using *Number Line* in the classroom. We worked closely with four classroom teachers to develop and implement *Number Line* with second graders over three days. In each class, the introduction on the first day was quick – less than 10 minutes. The microworld was projected on to the whiteboard, and each teacher had children read the first exploration (see Figure 1), and then illustrated this first instruction by dragging the **+3** block into the scripting area. "Nothing happens! But if I *click* on it…" The teacher clicked, and children described what they saw. Clicking a block performs the indicated arithmetic, shows the corresponding movement on the number line, and labels the result. Then a child was called up to continue the exploration. "Show the class how to drag out another block and make it do something." In one class, a child pulled out the **+5** block, clicked on it, and the children then saw this (Figure 3). In all classes, teachers also demonstrated how to link blocks together to make a simple script.

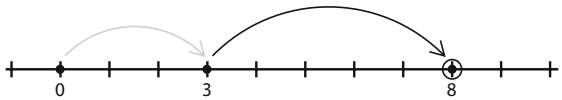

Figure 3: Introducing how blocks move the circle on the number line

After this minimal introduction, children returned to their desks to work on this first exploration, exploring the blocks and getting familiar with movements on the number line. Children then moved on to puzzles posing specific problem-solving challenges; below we share some of their approaches.

Lucia. Lucia was working on the puzzle *Make a script that starts at 0 and ends at 4*. The circle was at zero; she dragged out a **+3** block, clicked on it, saw that she landed on 3 and realized, "I need to jump one more space to get to 4." Instead of starting over, she pulled out another **+3** and landed on 6. This was too far—she still wasn't where she wanted to be—so she added the **–5** block to land on 1. Then she knew what she needed to do. She added **+3** and jumped to 4. Aha! Puzzles like these gave children experience relating jumps along the number line to small calculations such as, 3+3, 6-5, 1+3. As they worked on the puzzles, they monitored their own progress and experimented with different strategies.

***Estela*.** Estela was new to her class and new to the United States, her family having arrived recently from Guatemala. She used the Spanish text and audio, clicking on the puzzle to have it read aloud: "Haga un script que comience en 0 y termina en 6" (*Make a script that starts at 0 and ends at 6*). She started by pulling out **+5**, then adding **+3**. She ran her script, and realized it went too far. She looked for a **–2** block, but did not have one. She abandoned that script and pulled out two **+3** blocks to make a new script, perhaps remembering that 3 and 3 make 6.

Estela moved to the next puzzle, *Make a script that starts at 0 and ends at 2*. Like many children, she pulled out **+3** because she knew that 3 was close to 2. That jump went too far! She tried subtracting with a **-3**. That got her back to 0. She used another **-3**. She was on the other side of zero, but she saw exactly where the circle was on the numbert line and didn't get deterred by the negative sign. She used her finger on the screen to count from -3 to 2. 5! She added **+5** to her

script. She got it! She did a quiet dance in her chair and got the attention of Ivonne next to her. Ivonne was excited to see Estela's solution and showed Estela hers. Ivonne's script had taken the circle off the screen to the right and then back to 2 (see Figure 5). The girls excitedly explained their solutions to each other and tried to figure out how different scripts could have the same result. This example illustrates how solving puzzles through programming can support children's agency by easily making their mathematical thinking visible, enabling them to compare and discuss solutions.

Figure 5: Ivonne's solution

No puzzle in the microworld refers to negative numbers, but children often arrived there by accident, as Estela did. Children generally knew how to get back to "ordinary numbers," and many could predict which number they would arrive at. No further instruction was needed at this point—children were building informal experience with these numbers in ways that would support formalization in later grades.

Jay. Puzzles such as *Make a script that labels 1, 4, 7, and 10* gave children practice with the idea of repeated addition, foreshadowing multiplication: I'm adding 3, four times. Jay solved this puzzle with a simple script, built step by step: first, using the **start at 1** block, then recognizing quickly that a **+3** would move his circle to 4, and so on: **+3**, **+3**, **+3**, **+3.** Seeing this pattern motivated his interest in the **repeat** block. He rebuilt his script using **repeat**, with some trial and error ("How many times should it repeat?") (see Figure 4). But he was curious – what if he repeated more? He modified his script, choosing the maximum number of repeats (10). He

was thrilled to discover that the circle kept jumping by 3, off the screen where he could not see it – but he could monitor where it landed (31) using the **current number** indicator. He wondered, "Could I go to 1000?" He modified his script to start at 500 (the maximum number allowed) and ran it. He landed on 530. He took out the **+3** in the **repeat**, put in **+5** and re-ran the script – and landed at 550. He added another **+5** inside the **repeat**: 600. He was getting closer! He kept adding **+5** to **repeat** until he landed on 1000, with great satisfaction.

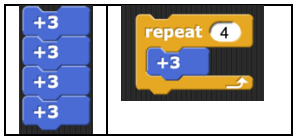

*Figure 4: Using **repeat***

Many children embraced **repeat** and used it to explore number patterns beyond what the puzzles required. *Number Line* does not directly pose problems like "*Start at 0 and jump to 1000*", but it does allow for easy and rapid experimentation in a way that pencil and paper cannot. We frequently saw children create and solve puzzles that were mathematically meaningful to them – developing their identities as *doers* of mathematics.

Kesha. Over the course of the lessons, children encountered more challenging puzzles. Kesha was working on *Make a script that lands on **all** the numbers from 0 to 10.* Like many children, she had already experimented with combining 3s and 5s, and she first thought about which numbers she knew how to make. She started by using **+3** three times to move from 0 to 3 to 6 to 9. She then added **-5** to jump back to 4. She recognized that adding **+5** would just take her back to 9, but **+3** would jump to 7, a number she hadn't yet landed on.

After some experimenting, Kesha commented, "What I really need is a 1." She was determined to approach it in a systematic way: starting from 0, then jumping to 1, 2, and so on. She built the script **+3**, **+3**, **–5** and tried it. She was delighted to see that the circle just one mark

on the number line, just as she had predicted. She clicked on her script again, and watched the circle jump to 2. “It’s at 2. Now I can just click it again.” She kept clicking as her circle jumped to 10 and beyond, with her jumps creating an intriguing and regular pattern. This is a wonderful informal example of reasoning by mathematical induction…from a 7-year-old! (See Figure 6.) During the class discussion, she enthusiastically shared her strategy and explained her reasoning. Kesha’s confidence in developing, using, and discussing her own strategy illustrates how she is building her mathematical competence and agency (NCTM, 2020).

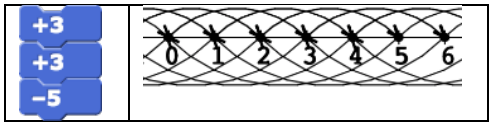

Figure 6: Kesha’s +1 script

Conclusion. Across classrooms, the microworlds supported robust child-to-child and child-to-teacher interaction. Children worked largely independently to solve puzzles; each child had direct experience running scripts and seeing their results. But the puzzles could be solved multiple ways, and children were curious about each other’s solutions. For each puzzle, children tracked: “I solved it” and “I showed someone” on a checklist. This simple device supported a lively dynamic of discussing solutions with tablemates and teachers. Teachers also used the end of each lesson for whole-class discussion of different approaches to solving a selected puzzle.

We were encouraged by children’s high level of engagement and persistence in problem solving in the microworld. Critical to the success of these lessons was how the microworld supported core mathematics content. Using *Number Line*, children were building fluency in adding and subtracting within 20 and predicting results mentally. They were gaining plentiful experience exploring number magnitude when they saw that the distance from 1 to 4 is the same as 4 to 7, and 7 to 10, and 5 to 8, and 8 to 5. Children experimented with the “any order” principle (the commutative and associative properties) of addition, noting that sequencing the

blocks in varying orders did not change the numerical result. They were also learning about how number lines themselves work, developing an understanding of the representation through repeated use.

Children were learning to express their mathematical thinking by communicating with the computer in a precise way, building scripts and running them. The computer "served as the tutee" (Taylor; 1980; Cullen et al., 2020) – the children were teaching the computer what to do, talking to it in a language it understands (in this case, Snap*!*). While their results were not always what they intended or expected, the microworld was responsive to children's actions and behaved with precision, allowing them to test and revise their conjectures.

Our experiences suggest there is potential for children to gain from curricular tools that embed programming into core mathematics instruction, supporting access to critical content. The examples in this article demonstrate how programming, when integrated into mathematics, can promote high levels of engagement in mathematical ideas, support problem solving and reasoning, and facilitate mathematical discourse. Additional research is needed to determine how use of these tools would affect children's learning over time.

More details about *Number Line* and related microworlds, including teacher guides and instructional videos, can be found at https://elementarymath.edc.org/. *Number Line* is freely available for use online at **https://go.edc.org/MW-number-line.**

Acknowledgements. We want to thank the children, teachers, coaches, and school leaders who made this work possible, with special appreciation to the Plympton Elementary School in Waltham, Massachusetts.

CCSSM 2.MD.B.6. Represent whole numbers as lengths from 0 on a number line diagram with equally spaced points corresponding to the numbers 0, 1, 2, ..., and represent whole-number sums and differences within 100 on a number line diagram.
CCSSM 2.OA.B.2. Fluently add and subtract within 20 using mental strategies. By end of Grade 2, know from memory all sums of two one-digit numbers.
CSTA K12 CS Standards 1A-AP-10. Develop programs with sequences and simple loops, to express ideas or address a problem.
CSTA K12 CS Standards 1A-AP-14. Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.

Role(s) of Technology

Conveyance Technology Uses	Mathematical Action Technology Uses
• *Encouraging Collaboration: Encouraging collaboration around mathematical problems.* • *Monitoring and Assessing: Monitoring and assessing mathematical learning.*	• *Serving as Tutee: Decomposing, abstracting, and encoding mathematical procedures and processes.* • *Promoting Cycles of Proof: Creating, testing, revising, and proving mathematical conjectures.*

Teaching Practices

- Implement tasks that promote reasoning and problem solving.
- Use and connect mathematical representations
- Pose purposeful questions
- Support productive struggle in learning mathematics
- Elicit and use evidence of student thinking

References

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K–6 computational thinking curriculum framework: Implications for teacher knowledge. *Educational Technology & Society*, 19(3), 47–57.

Bers, M. U., Govind, M., & Relkin, E. (2022). Coding as another language: computational thinking, robotics and literacy in first and second grade. In *Computational Thinking in PreK-5: Empirical Evidence for Integration and Future Directions*, 30–38. https://doi.org/10.1145/3507951.3519285.

Bers, M. U. (2018). Coding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom. Routledge Press. https://doi.org/10.4324/9781315398945.

Code.org (n.d.). About Us. Retrieved December 31, 2022, from https://code.org/about.

CSforALL (n.d.). Computer Science for All. Retrieved December 31, 2022, from https://www.csforall.org/about/.

Cullen, C., Hertel, J., & Nickels. M. (2020) The roles of technology in mathematics education. *The Educational Forum*, 84 (2), 166–178. https://doi.org/10.1080/00131725.2020.1698683

Cuoco, A., & Goldenberg, E.P. (1996). A role for technology in mathematics education. *The Journal of Education,* 178 (2), 15–32. https://doi.org/10.1177/0022057496178002

Dick, T. P., & Hollebrands, K. F. (2011). *Focus in high school mathematics: Technology to support reasoning and sense making*. NCTM.

Expanding Computing Education Pathways [ECEP]. (n.d.). ECEP Mission. Retrieved December 31, 2022, from https://ecepalliance.org/about/mission/

Fofang, J. B., Weintrop, D., Walton, M., Elby, A., & Walkoe, J. (2020). Mutually Supportive Mathematics and Computational Thinking in a Fourth-Grade Classroom. *International Conference of the Learning Sciences* conference proceedings, 1389–1396. https://doi.dx.org/10.22318/icls2020.1389

Goldenberg, E.P., Carter, C.J. (2021). Programming as a language to express and explore mathematics in school. *British Journal of Educational Technology*, 52, 969–985. https://doi.org/10.1111/bjet.13080

Goldenberg, E.P., Carter, C.J., Mark, J., Reed, K, Spencer, D., & Coleman, K. (2021). Programming as language and manipulative for second-grade mathematics. *Digital Experiences in Mathematics Education*, 7, 48–65. https://doi.org/10.1007/s40751-020-00083-3

Logo Foundation. (n.d.). *Logo history.* Retrieved December 31, 2022 from https://el.media.mit.edu/logo-foundation/what_is_logo/history.html

McCulloch, A., Lovett, J., Dick, L., & Cayton, C. (2021). Positioning students to explore math with technology. Mathematics Teacher: Learning and Teaching PK–12, 114 (10), 738–749. https://doi.org/10.5951/MTLT.2021.0059

National Council of Teachers of Mathematics (NCTM). (2020). *Catalyzing Change in Early Childhood and Elementary Mathematics*. NCTM.

National Council of Teachers of Mathematics (NCTM). (2014). *Principles to actions: ensuring mathematical success for all*. NCTM.

New York City Department of Education [NYCDOE, n.d]. *Welcome to CS4All*. Retrieved December 31, 2022 from https://sites.google.com/schools.nyc.gov/cs4allnyc/.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Sherwood, H., Yan, W., Liu, R., Martin, W., Adair, A., Fancsali, C., Rivera-Cash, E., Pierce, M., & Israel, M. (2021). Diverse approaches to school-wide computational thinking integration at the elementary grades: A cross-case analysis. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education,* 253–259. https://doi.org/10.1145/3408877.3432379

Taylor, R. P. (1980). Introduction. In R. P. Taylor (Ed.), *The Computer in the School: Tutor, Tool, Tutee* (pp. 1–10). Teachers College Press.