Journal of Pedagogical Research
Volume 7, Issue 2, 2023
https://doi.org/10.33902/JPR.202318636

Review Article

When preservice and inservice teachers join forces: A collaborative way to support the enactment of new coding curricula in mathematics classrooms

Laura Broley[1], Chantal Buteau[2] and Jessica Sardella[3]

[1]*Brock University, Canada (ORCID: 0000-0003-4268-9603)*
[2]*Brock University, Canada (ORCID: 0000-0001-9401-7011)*
[3]*Brock University, Canada (ORCID: 0009-0000-1668-2569)*

The importance of computational thinking skills in mathematics has been recognized in educational research for a long time. More recently, this recognition has materialized in formal international recommendations (e.g., by PISA's 2022 Mathematics Framework) and in national or provincial curricular reforms (e.g., in France, Sweden, and Canada) that promote the incorporation of coding in mathematics classrooms. This has led to opportunities as well as challenges for mathematics teachers, and a pressing need for work on teacher training. To contribute to this emerging area, we report on a professional development experience in which 25 inservice teachers collaborated with 36 preservice teachers to plan, implement, and reflect on the implementation of coding-based mathematics activities (using Scratch or Python) with Gr. 5–9 school students. Teachers' reflections are shared as insights gained through the experience, which may be of interest to other teachers or policy makers engaged in the implementation of coding in school subjects such as mathematics. With other researchers and teacher educators in mind, participating teachers' reflections are also used as a springboard to evaluate the reported training approach, discuss the approach in the context of existing literature, and provide some perspectives for the future.

Keywords: Coding; Computational thinking; School mathematics education; Teacher education; Teacher professional development; Curriculum enactment

Article History: Submitted 24 October 2022; Revised 8 March 2023; Published online 10 June 2023

1. A Challenge Facing Teachers around the World

During the last two decades, there has been a worldwide shift towards the explicit inclusion of "computational thinking" (CT)[2] as an essential competency of the 21st century. This has included curricular revisions in K–12 education in numerous countries, with varying approaches to integrating CT (Bocconi et al., 2016; Bocconi et al., 2018; Dagienė et al., 2019; Gannon & Buteau,

Address of Corresponding Author

Chantal Buteau, PhD, Brock University, Faculty of Mathematics & Science, 1812 Sir Isaac Brock Way, St. Catharines, ON, L2S 3A1 Canada.

✉ cbuteau@brocku.ca

How to cite: Broley, L., Buteau, C., & Sardella, J. (2023). When preservice and inservice teachers join forces: A collaborative way to support the enactment of new coding curricula in mathematics classrooms. *Journal of Pedagogical Research, 7*(2), 21-40. https://doi.org/10.33902/JPR.202318636

[2] There are many different definitions of CT used in educational literature and definitions vary even among the works cited in this paper. Aligning with our focus on programming is the following, simple, and commonly cited definition: CT comprises "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Cuny et al., 2010, cited in Wing, 2010, p. 1).

2018; Wu et al., 2020). One commonality is the teaching of computer programming (or "coding")[3] to students starting from a young age (Rich et al., 2019). Although there seems to be a consensus that CT is much more than programming, the latter is typically seen as an important, engaging, and productive way of learning and applying CT (Grover & Pea, 2018). Programming itself has also been argued to be a potentially useful tool – for thinking, learning, and doing (Guzdial, 2019), or for reasoning about the world, engaging in creative expression, and participating meaningfully in computational spaces (Tissenbaum et al., 2021) – for any student, not only those who will eventually study computer science (CS) or become professional software developers.

CT – and, in particular, programming – has often appeared in curricula as an isolated objective, rather than being integrated to enrich existing subjects (Dagienė et al., 2019; Gadanidis et al., 2017). A recent survey of 313 teachers from 23 countries found that almost double the number of teachers (212 vs. 108) teach coding as a standalone subject rather than as an integrated topic (Rich et al., 2019). This said, the thoughtful integration of programming with other school subjects, such as mathematics, science, language, arts, social studies, and music, may have bidirectional benefits for learners, enriching the learning of both the subject and the programming (Grover & Yadav, 2020); and many research studies have demonstrated this in STEM subjects (Grover & Pea, 2018; Weintrop et al., 2016). Some researchers highlight natural, deep, and historical connections between programming and mathematics in particular (e.g., Gadanidis et al., 2017; Grover & Yadav, 2020). Rich et al. (2019) report that when coding was taught as an integrated topic, it was, "perhaps expectedly," most often integrated with mathematics (p. 318). That incorporating programming into mathematics education has many potential benefits is not a new idea (Benton et al., 2017; Papert, 1980). The recent renewed focus on CT has nevertheless brought a new life to the idea (Grover & Yadav, 2020), further supporting the possibility of it coming to life in classrooms.

More recently, there has been increased institutional-level support for the combined teaching of programming and mathematics in the mandatory education for all students. Several jurisdictions have formally integrated programming into mathematics curricula starting in Gr. 1, including France in 2016 (Gueudet et al., 2017), Sweden in 2018 (Vinnervik, 2022a), and the province of Ontario (Canada) in 2020. For instance, in Ontario – the context where we work – the Ministry of Education introduced a new collection of "coding" expectations as part of the Algebra strand in Gr. 1–9 Mathematics (2020, 2021). All students are expected to learn to use coding concepts and skills to solve problems across mathematical topics and to create computational representations of mathematical situations, by reading, altering, writing, and executing code. Specific concepts and skills are suggested to be learnt in a progression (e.g., sequential events in Gr. 1, nested events in Gr. 4, and subprograms in Gr. 7). At this time, the Gr. 10–12 Mathematics curricula have not yet been revised (including to possibly align with the new Gr. 1–9 coding expectations). On a larger scale, recent changes to PISA's Mathematics Framework explicitly encourage participating countries to reflect on the role of CT – "conceptualised as defining and elaborating mathematical knowledge that can be expressed by programming" (p. 12) – in mathematics pedagogy and curricula: From 2022 onwards, educational systems around the world will be evaluated on how well they develop students' CT skills as part of their problem-solving practice in mathematics (OECD, 2018). Adapting to new curricular expectations can be challenging for teachers (Sentance & Csizmadia, 2017; Vinnervik, 2022a, 2022b). Like Vinnervik, we assume that it may be particularly challenging in the case of teaching programming, which is still considered a grand challenge in computing education (Caspersen, 2018). One reason for this is that programming is a

[3] In the same vein as CT, there exist varying definitions and distinguishable interpretations of "programming" and "coding" in educational literature (see Armoni, 2016 for an example of a discussion about this issue). In this paper, we use the two words interchangeably to refer to the process of creating a computer program, which comprises the often-intertwined steps of designing the program (including the algorithms involved), implementing the program (including writing code), and validating that the program works. This choice resulted naturally from the merging of our previous work (in which we have typically used the term "programming" in reference to a cycle of design and validation; see, e.g., Buteau et al., 2016) and the new school curriculum in our context (which uses the term "coding"). Although definitions may differ across the cited works, we consider the cited ideas as applicable to our chosen definitions.

creative process, much like writing (ibid.): its potential for addressing a range of meaningful projects depends on concepts, practices, and perspectives (Brennan & Resnick, 2012) that cannot be learned overnight by teachers, many of whom are starting with little to no programming experience. A study involving 1197 teachers in Finland, Mainland China, Singapore, Taiwan, and South Korea, all of which have experienced recent CT-related curricular revisions, concluded that "one of the most striking findings that concern all the education systems is the fact that the majority of the teachers are not yet competent in any coding language" (Wu et al., 2020, p. 32). Similarly, a survey of 1342 primary teachers in Lithuania (representing 20% of all primary teachers in the country), where a new integrated informatics curriculum is being implemented starting in Gr. 1, found that teachers feel weakly or not at all prepared to teach algorithms and programming (Dagienė et al., 2019). In the case of an integrated curriculum, an additional component of the challenge is that teachers are expected to teach not only programming, but also how it can be used as a productive tool in other subject areas. In Sweden, for example, an initial study involving 133 teachers found that although the teachers had positive attitudes towards programming and its integration in mathematics, they did not feel well prepared for teaching programming in mathematics (Misfeldt et al., 2019).

Most scholars cited above identify the need for creating specific training to support teachers in gaining the skills, confidence, and didactic approaches required to enact new CT-enriched curricula. In the beginning of 2022, 25 inservice teachers and 36 preservice teachers worked in teams to prepare and implement integrated coding and mathematics activities (using Scratch or Python) in Gr. 5–9 classrooms in Ontario. This paper provides a report of this collaborative approach as one possible way of addressing the need for training for both preservice and inservice teachers. After reviewing some emerging literature on (mathematics) teacher training in CT (Section 2), we describe our specific approach and how it emerged in our context (Section 3). We then present reflections from participating teachers provided mainly during a post-implementation collective reflection session (Section 4), which allows us to offer an evaluation of the approach (Section 5). We end with some concluding remarks, including some critical perspectives and considerations for future iterations of the collaboration (Section 6). We expect our report to be useful to other educators providing training to preservice and/or inservice teachers. The reflections made by teachers participating in our collaborative approach may also be of value for other teachers or policy makers involved in integrating coding in mathematics education.

2. Literature on (Mathematics) Teacher Training in CT

Systematic literature reviews suggest that our understanding of (mathematics) teacher training in CT is still emerging. One study of articles published between 2006 and 2017 that evaluate didactic activities developing CT and mathematics found that only three (out of 42) addressed teacher training (Barcelos et al., 2018). Similarly, in reviewing research on preservice and inservice K–6 training for teaching computing, coding, robotics, or CT published in 2008–2018, Mason and Rich (2019) identified only 21 relevant papers, most of which appeared between 2016 and 2018, and only eleven of which took an integrated approach (vs. teaching CS as its own content area). Already in these papers, important commonalities were identified. For instance, most studies present positive results in terms of assisting teachers in overcoming knowledge, attitude, and efficacy barriers; but there is a greater focus on developing attitudes towards and knowledge in CS, rather than pedagogical knowledge. One early study also pointed to the importance of future training developing teachers' understanding of CT within the subject matter they teach:

> Unless their knowledge is developed in that context, teachers may only gain an "abstract" understanding of CT. As a result, their knowledge will remain inert and they will be unable to incorporate it into their teaching [Brown et al. 1989]. (Yadav et al., 2014, p. 14)

Over the last five years, articles have emerged presenting case studies of preservice teacher training in CT and related pedagogy in the context of mathematics. Researchers from Western University in Ontario, for example, describe specially designed courses for preservice elementary

(K–6) mathematics teachers (Gadanidis et al., 2017) and preservice secondary (Gr. 7–12) mathematics and science teachers (Araujo et al., 2019). Both courses focussed on coding as a tool for developing CT and provided preservice teachers with concrete lesson ideas; for instance, through school-level mathematics activities with different coding technologies, video examples of authentic coding in mathematics classrooms, and relevant literature. Preservice teachers also completed collaborative work and individual reflections, which evidenced their learning about the meaning, affordances, and integration of CT in mathematics education. Gadanidis et al. (2017) suggest that some preservice teachers may have transferred their learning into their practicum placements; however, their evidence was only anecdotal.

Suters (2021), also reporting on a case of preservice science and mathematics teacher training (within the United States), aimed to fill a gap in the literature by extending teachers' learning to include practice teaching CT in authentic contexts. In four first-semester courses taken by future elementary teachers (science methods, mathematics methods, instructional technology, and a 60-hour practicum placement), the author planned a coherent set of CT interventions, where CT was considered through a broad range of activities (unplugged data collection, problem solving using 3D printers, robotics, and coding). In addition to engaging in CT, readings, and reflections, preservice teachers eventually used their knowledge of CT to run stations at "STEM nights" at local schools and to team-teach CT-based robotics or makerspace lessons in Gr. 3–5 mathematics or science classes. Suters (2021) reports improvements in preservice teachers' self-efficacy, views, and pedagogical knowledge. Suters (2021) also notes that inservice teachers taught with each preservice teacher team and conjectures that preservice teachers' lessons could "serve as models for inservice teachers as ways in which they could teach content and incorporate CT … within their classrooms" (p. 366); the inservice teachers, however, were not the focus of the study.

Ketelhut et al. (2020) suggest that a significant portion of literature has focussed on preservice teachers:

> While future educators are a logical avenue for long-term change … there is also a need to prepare current teachers to integrate CT into their classrooms—especially if we consider the ubiquity of computing as not only growing but accelerating. (p. 175)

To address the gap, Ketelhut et al. (2020) present a year-long PD experience for inservice Gr. 1–5 science teachers, including a workshop (to develop a base knowledge of CT) and participation in an inquiry group with preservice teachers and researchers, culminating in an activity of collaboratively designing CT-infused science lessons. Ketelhut et al. (2020) report that the inservice teachers developed positive views and impactful lessons, but also experienced several struggles (e.g., understanding how CT could best fit with their curricula, finding resources and support) and they did not naturally include computation (e.g., programming) within their lessons.

Studies specific to inservice mathematics teacher training are also appearing. Similar to Ketelhut et al. (2020), Reichert et al. (2020) and Kelter et al. (2021) describe training experiences for inservice teachers involving workshops (covering CT concepts) and the collaborative design (under the guidance of or with researchers) of CT-infused activities, which are then implemented by the teachers. Reichert et al. (2020) present an 8-month (32-hour) continuing education course involving Gr. 6–9 mathematics teachers, while Kelter et al. (2021) study a 4-week (20-hour) summer program involving high school mathematics and science teachers. As usual, the training was found to have positive impacts (on teachers' understanding and/or confidence in CT). This said, Reichert et al. (2020) report that the teachers in their study still felt the need for assistance from a CS specialist in implementing activities involving programming. In a similar vein, Kelter et al. (2021) highlight how dependency arising in collaborative lesson design may affect teachers' CT skill development and implementation: In particular, some teachers (with less experience in programming) had their collaborator (a researcher) create the computational tools for their lessons, while they focussed on pedagogical aspects, which seemed to limit their opportunities for developing CT skills and discourage them from focussing on programming while implementing their lessons.

In sum, literature on mathematics teacher training in CT is emerging, with a growing collection of case studies documenting various effective approaches. Recent studies seem to agree on the importance of making such training specific to the subject matter taught by participating teachers, as well as providing opportunities to not only learn about but also practice teaching CT. Although some reported approaches may target both preservice and inservice teachers (e.g., Ketelhut et al., 2020; Suters, 2021), it seems that papers typically address one of the two populations. Mason and Rich (2019) explain: "Although preservice and inservice training may be similar, the needs of preservice teachers differ from those of inservice teachers" (p. 794). We hypothesize that an approach bringing together preservice and inservice teachers may nevertheless be productive for meeting some of the needs of both populations, while also supporting the enactment of new CT-related curricula. In the next section, we describe one such approach (Section 3.2) and how it came to be in our context (Section 3.1).

3. Our Collaborative Approach to Teacher Training

3.1. Contextual Background

In Canada, the curricula that govern what is taught and learned in schools are determined on a provincial or territorial level, and approaches to integrating CT can vary (Gannon & Buteau, 2018). In the province of Ontario, CT-related content – in particular, computer programming – has been integrated in optional secondary school (Gr. 9–12) courses (e.g., in CS) since 1966 (Floyd, 2022). As outlined in Section 1, the Ontario Ministry of Education only recently (2020/2021) made coding an official part of the mandatory education for all students, locating it within the Gr. 1–9 Mathematics curricula.[4]

The mathematics teacher training that we describe in the next section has roots in a collaborative professional development initiative that originated in the 2017–18 school year, prior to the curricular reform mentioned above; that is, prior to the formal inclusion of coding in Gr. 1–9 Mathematics. Framed and funded by the provincial Mathematics Knowledge Network (MKN),[5] the initiative was led by individuals across institutional boundaries (school and university) who shared a belief that programming can enrich students' mathematics learning and who were in positions aimed at supporting mathematics teachers: Two numeracy consultants (Laura Cronshaw and Jeff Martin) who worked with inservice teachers in the Niagara Catholic District School Board and a professor (Buteau) involved in training preservice mathematics teachers at Brock University. More specifically, Buteau was teaching preservice teachers in the context of three university-level mathematics courses called Mathematics Integrated with Computers and Applications (MICA) I, II, and III. Since 2001, the MICA courses have engaged mathematics majors and co-majors (including the preservice mathematics teachers) in learning to use programming in pure or applied mathematics investigation projects: to explore mathematics concepts (e.g., dynamical systems, Monte Carlo integration), conjectures (e.g., about prime numbers or hailstone sequences), and real-world applications (e.g., traffic flow, population dynamics, the stock market; see, e.g., Buteau et al., 2019; Buteau et al., 2016; Buteau et al., 2015). In the 2017–18 school year, Buteau created a new version of the MICA III course dedicated to the preservice mathematics teacher population. This version included a new final project, developed in collaboration with Cronshaw and Martin, in which the preservice teachers would have the opportunity to work with inservice teachers to explore how programming-based mathematics learning (what they had experienced in the MICA courses) might be productively transferred into school mathematics classrooms.

In light of the 2020/2021 curricular reform, the MICA III course for preservice teachers was revised (Broley et al., 2023) and we came to see the final collaborative project as a sort of training capable of building both preservice and inservice teachers' capacities in integrating coding into their mathematics teaching (and, by extension, supporting the enactment of the new curriculum).

[4] The Ontario Ministry of Education has also recently (2022) revised the Gr. 1–8 Science and Technology curriculum to include coding.

[5] See http://mkn-rcm.ca/ for more information, including the MKN's guiding principles, which also guided our training approach.

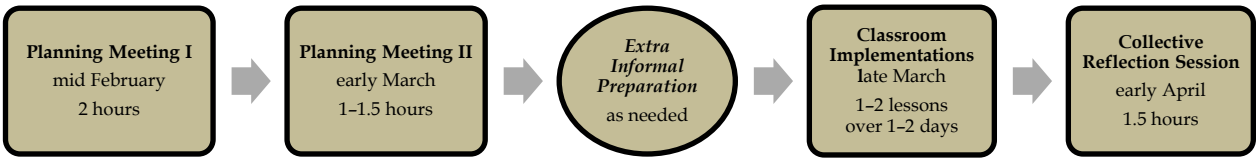
In this paper, we focus on the final project that took place in 2022 as representative of our collaborative training approach.

3.2. Details of the Approach

Our collaborative training approach invited preservice and inservice teachers to work together in teams to prepare and implement coding-based activities (using Scratch or Python) for mathematics learning in school classrooms in Ontario (Canada). In 2022, the collaboration comprised 18 teams involving 36 preservice teachers (two per team) and 25 inservice teachers (one or two per team, and one team involving four),[6] working under the direction of four facilitators (Cronshaw, Martin, Buteau, and Broley). The inservice teachers were teaching mathematics in Gr. 5, 6, 7, 8, or 9 at schools within the Niagara Catholic District School Board and agreed to participate through recruitment by numeracy consultants Cronshaw and Martin. Aligning with the international picture described in Section 1, many of the inservice teachers expressed having a beginning fluency in coding (i.e., limited experience and confidence in coding), though there was a range of skills indicated. The preservice teachers were third- or fourth-year students at Brock University taking the MICA III course taught by Buteau as part of an education and mathematics program providing certification for teaching Gr. 7–12. In contrast with the inservice teacher group, all preservice teachers had learned how to use coding to conduct mathematics investigations within the MICA courses described in Section 3.1.

Figure 1 offers a structural overview of the collaboration that took place in February to April of 2022. It was structured through two planning meetings involving all collaborators (on February 16 and March 9), the classroom implementations by pre/inservice teacher teams of coding activities for learning mathematics (during the week of March 28), and a post-implementation collective reflection session (on April 6). We provide a brief description of each of these elements below.[7] The inservice teachers were compensated for their time participating in the two planning meetings and the collective reflection session, which took place outside of school hours. For the preservice teachers, the collaboration (including all elements in Figure 1) constituted a significant component of their MICA III course, counting for 30% of their final grade. The facilitators of the collaboration met on several occasions prior to February 2022 for preparation and organization purposes.

Figure 1
The structure of our collaborative training approach, adapted from Sardella et al. (2023)



The first planning meeting took place online[8] and lasted two hours. After the collaboration was introduced by the facilitators, teacher teams met in breakout rooms to introduce themselves and start planning their implementations. Inservice teachers were invited to describe their classroom culture and needs, in hopes that the experience would be personally and professionally meaningful to them (a key guiding principle of the MKN). Teams also discussed the coding and mathematics topics they could address and whether they wanted to use or adapt an activity from

[6] The initial collaboration mentioned in Section 3.1 (in 2018) involved only five preservice teachers and four inservice teachers. We suspect that the 2020/2021 school curriculum revisions are partially responsible for the increased participation rate in 2022.

[7] For more details, including guidelines and preparatory activities for each of the planning meetings and the collective reflection session, see http://mkn-rcm.ca/niagara-catholic-brock-u-collaborative-coding/ For an elaborated example of how one teacher team experienced each component of the collaboration, see Sardella et al. (2023).

[8] The online setting was chosen due to COVID-related restrictions and seemed to simplify the coordination of meetings among teachers coming from different schools. We would have liked to conduct these meetings in-person since it would have encouraged interactions among teacher teams and made it easier for facilitators to oversee their work.

one of the suggested resources (developed specifically for the Ontario curriculum), or to create a new activity.[9]

The second planning meeting also took place online and lasted between one hour and 90 minutes. Each team met to further discuss and refine their chosen activity, and to plan how it would be implemented in the classroom, including certain logistical elements (e.g., timing, classroom set-up) and the individual roles of team members. After a team finished refining their activity, one of the facilitators met with them to provide feedback. Many teams needed to engage in additional preparation outside the planned meeting times, which often required further team meetings, work on the activity by the preservice teachers (who typically took charge of leading the lessons), and preparation of school students by the inservice teachers.

The classroom implementations took place in-person (16 teams)[10] or online (two teams). Depending on the team, activities were implemented through one or two lessons over one or two days, and possibly in multiple classes (when the team involved more than one inservice teacher). When possible, one of the facilitators was present for observation and support. After the implementations, questionnaires were administered to preservice and inservice teachers to prompt reflections on their implementations in preparation for a collective reflection session.

The collective reflection session was held online for 90 minutes. After a brief introduction, the pre/inservice teacher teams were grouped in six breakout rooms (three teams per room), each with a moderator. Discussions in these groups, for over 50 minutes, focused on answering three questions based on the classroom implementation experiences:

1. What were the major learning outcomes when integrating coding in the mathematics classroom?
2. What were the major challenges when integrating coding in the mathematics classroom?
3. What recommendations would you give to teachers for their preparation or implementation of such an activity?

Each group used a Jamboard to record their answers in the form of brief comments and different colours were used to keep track of which team contributed which comment. For the remainder of the session, key ideas from each group were shared in a whole group setting.

During the collective reflection session, all attending teachers (36/36 preservice teachers and 20/25 inservice teachers) agreed to have their Jamboard comments used for the purposes of disseminating the collaboration to other practitioners. To give an organized overview of what teachers said, their comments were read and sorted with the aim of identifying themes. These themes were separated depending on whether they related to teachers' experiences of implementing coding in mathematics classrooms or teachers' experiences of the collaboration, and they were further grouped into benefits, challenges, or recommendations, aligning with the guiding questions listed above. Although the above questions do not directly probe into teachers' views of the collaborative approach, such views were present in some of the Jamboard comments. To enhance the collection of themes identified and the corresponding collections of illustrative comments, we also explored post-implementation questionnaire responses made by inservice teachers who gave their consent, as well as publicly available video snippets[11] from interviews we conducted with three inservice teachers after the collaboration was completed. Section 4 presents teachers' reflections organized through the identified themes. In Section 5, we then use these

[9] Teams were free to choose which topic(s) to address, which programming language(s) to use, and the structure of their activity according to their interests and needs. This resulted in a range of mathematics topics (e.g., similar shapes, financial mathematics, the Pythagorean theorem). All teams used Scratch or Python, likely due to a combination of factors: e.g., these are the commonly used programming languages in Ontario schools and related resources, and the preservice teachers had used Python and were introduced to Scratch in their MICA courses. In a similar vein, many activities followed a "use-modify-create" structure (Waite & Grover, 2020) due to the resources used and/or the fact that it had been introduced to preservice teachers in the MICA III course (though there were variations in the extent to which the classroom implementations actually reached the "create" stage). For some examples of topics addressed and activities used, see http://mkn-rcm.ca/niagara-catholic-brock-u-collaborative-coding/

[10] In three of these teams, one preservice teacher was forced to participate online due to late screening reference checks (required when going into schools) or COVID isolation requirements.

[11] See http://mkn-rcm.ca/niagara-catholic-brock-u-collaborative-coding/

themes to support an evaluation of our collaborative approach to teacher training and curriculum enactment.

4. Reflections by Teachers Participating in our Collaborative Approach

We present reflections made by preservice and inservice teachers participating in our collaborative approach in two parts: first (Section 4.1) concerning the implementation of coding in mathematics classrooms, and second (Section 4.2) concerning the collaborative approach.

4.1. Teachers' reflections on implementing coding in mathematics classrooms

Table 1 summarizes benefits, challenges, and recommendations that were recognized by preservice and inservice teacher teams participating in our collaborative approach concerning the implementation of coding in mathematics classrooms. These are further described, with the support of illustrative quotes, throughout Sections 4.1.1 (benefits), 4.1.2 (challenges), and 4.1.3 (recommendations). For an example of how one teacher team experienced some of these, see Sardella et al. (2023).

Table 1
Benefits, challenges, and recommendations related to implementing coding in mathematics classrooms, based on teachers' reflections on their own implementation experiences while participating in our collaborative approach

Benefits
• Supporting multiple levels of learning • Encouraging (new) student leaders • Increasing student engagement and motivation • Inviting students to be involved in their learning (increasing student agency) • Providing students with a tool for learning mathematics concepts • Providing students with a tool for engaging in mathematical practices • Allowing students to learn mathematics in new and different ways • Enabling students to gain familiarity with coding languages, skills, and uses
Challenges
• Timing the lesson • Ensuring students stay engaged and on task • Addressing students' differences • Responding to students' needs • Connecting and combining mathematics and coding • Supporting students who lack experience in coding • Handling technological issues • Working with classroom constraints
Recommendations
• Allow students to collaborate • Ensure students have sufficient knowledge and/or scaffolding • Do not limit coding tools to the computer • Highlight the conventions of coding • Give sufficient time for student exploration • Check-in on students • Create lessons that meet varying levels of need • Foster an environment that is accepting of errors and questions • Be prepared, but be adaptable (just go with it) • Acknowledge and accept that teachers are not experts • Make coding a consistent part of the classroom

4.1.1. Benefits

During the collective reflection session, some pre/inservice teacher teams commented on the potential of coding activities for *supporting multiple levels of learning*. For instance, one team suggested that "the coding was applicable to all levels of learning (beginner–advanced)" and there was an "entry point for all learners." Another team noted that "student engagement in coding was high, regardless of skill level in Scratch" and highlighted the creation of extension questions as "allowing students to excel in their learning if they had time." Related to this is the benefit of *encouraging (new) student leaders*: One teacher observed "students that grasped coding … become leaders in the classroom and helping their classmates"; another specified that implementing coding "allows students to succeed that sometimes struggle in other subject areas."

The potential benefits of coding for *increasing student engagement, motivation, and agency* were highlighted by many teams. Teachers' comments suggested that students were "having fun," that the new coding topic "sparked a lot of curiosity," that "students were motivated to try the challenges," and that "students were excited seeing how the code came together to work." Another collection of comments described coding as enabling students to take ownership of their work, explore, experiment, be creative, and share their work with others. One team, for example, mentioned their students' "positive response to creating their own code"; a second team pointed to students "being able to experiment with code" and developing "increased agency to create their own thing [that is] unique in its own sense"; according to a third, "students loved sharing their work with everyone."

Some teams' comments touched on benefits that are specific to mathematics, such as coding *providing students with a new tool for learning mathematics concepts* (either deepening their learning of previously learned concepts or learning new concepts) *or for engaging in mathematical practices* (such as graphing or problem solving). Indeed, many comments featured the specific mathematics concepts addressed by implemented activities. For example, one team said their activity served to "consolidate [students'] learning of the Pythagorean Theorem through a coding application." Another team suggested that their activity "ingrained in the students how perimeter and area are linked." A third team described their students as "using coding as a learning tool" for "balancing equations." Referring to the probability model students explored in their activity, a fourth team claimed that students gained a "better understanding of the math process; [they were] able to see it, see the math behind the problem." More generally, this team also hypothesized that "students might have learned how to use coding for mathematics. For example, word problems in a way that a computer can understand." Some teachers also recognized coding as *allowing students to learn mathematics in new and different ways*. One team said that "students got a chance to see math topics in more interactive ways," while another pointed to students' "idea generation and seeing how to solve problems in different ways, try[ing] different routes, [and] expand[ing]."

Perhaps not surprisingly, many teams alluded to the fact that implementing coding in mathematics classrooms entails the learning of coding itself, *enabling students to gain familiarity with coding languages, skills, and uses*. For instance, one team reported that students "became more comfortable with coding in Scratch" and "testing and debugging as a necessary coding practice." Another team explained that "students learned how to run code on Jupyter Lab … [and] [t]hey learned that Python has many real-life uses." Several teams brought up students' learning of skills related to "problem solving," "perseverance," or "trial and error." Several also indicated that students may have experienced important uses of coding: e.g., "Students got a chance to see how efficient code can be for exploring math topics!"

4.1.2. Challenges

When it came to reflecting on the challenges they experienced in implementing coding in mathematics classrooms, many pre/inservice teacher teams commented on issues related to *timing the lesson*. This includes difficulties in getting through everything planned and, in relation to this, determining how much time coding activities require. One team reflected that they "maybe

overestimated [students'] coding skills, [and] over-created activities for the amount of time it actually takes in the classroom." In a similar vein, another team said that "fitting in such an in-depth coding task … within 45 minutes was tough." Other teams highlighted difficulties related to progressing through a lesson in such a way that keeps students engaged, allows them to explore, and ensures their understanding. One team, for example, spoke about the "tension between exploration and getting through [the] lesson," while another explained that it can be difficult to "find a balance of how much time to give students on each aspect so students stay interested."

This last comment also relates to the challenge of *ensuring students stay engaged and on task*. Although coding was seen as increasing student engagement, some teachers noted that keeping students on track during the process of coding can be difficult. Several teams mentioned distractions related to coding, such as features of a programming language that may or may not be relevant for the task at hand. One team recounted that "students got distracted by the coding environment" (Scratch) and that there were "cats meowing everywhere." Other possible distractions were also mentioned: e.g., students "having too much fun" or encountering a break in a lesson. As one team explained: "We had a recess in between the lessons so regaining focus was a bit tricky."

Addressing students' differences was another, complex, challenge that emerged from numerous comments. Teams acknowledged the varying levels of "comfort," "confidence," "knowledge," "experience," and/or "interest" among students, in relation to both mathematics and coding. For instance, one team said that "some students had previous coding experience whereas others were using Scratch for the first time." Another team found that different students had a "different understanding of angles; so some need[ed] extra help." Several teams observed how differences among students could lead to different student responses and a "need to differentiate" by the teacher. One team suggested that some students were "maybe not ready" and needed "more time on Scratch," although they "did want to do the same thing as their peers." The same team also recalled that:

> Some [students] went with it … Others got frustrated … got stuck, let anger take over. Some kids love the frustration … Some [are] still wrapped up in the fact it was not working.

Two other teams observed similarly: "sometimes students who 'know more'/'are very confident' expect everything to be perfect and experience some stress, less willing to take risks"; "some students … took initiative with the exploration while some students desired high guidance and didn't want to try new things."

Closely connected to the challenge of addressing students' differences is the challenge of *responding to students' needs*. In reference to the "different levels of students in class," one team concluded: "So it could be difficult to ensure all student needs [are] met." Comments related to this challenge point to the difficulties of "trying to help many students at the same time," getting students to ask for help in the first place (since "students who were stuck often would not ask for help"), and using productive strategies for providing help (e.g., "with[out] giving the answer"). Some teams also pointed out that being responsive to students' needs requires the teacher to be flexible, sometimes re-evaluating and adapting their lessons. One team recognized this as "harder for planning purposes" but "better for a more suitable lesson that helps kids be successful."

Two challenges in Table 1 are specific to the subject matter: *connecting and combining mathematics and coding*, and *supporting students who lack experience in coding*. One teacher's comment illustrates nicely both challenges:

> One point the [preservice teachers] and I kept coming back to was are we teaching code using math or are we teaching math concepts using code. While we agreed it was the second point, the main hurdle was how do we teach a math concept using a method that students do not understand…

Implicit in this comment is the challenge, for teachers, of making connections between mathematics and coding, which is linked to determining the learning goals (e.g, coding, mathematics, or both) and supporting students in meeting them. Some teams described the challenge of getting students to understand both coding and mathematics: e.g., "ensuring students

understand coding concepts and its implementation of math within it … [to] see the math inside coding." One team alluded to the increased cognitive load that students may experience when combining coding and mathematics concepts (both of which may be new to them):

> Combining a new coding language and mathematics that was trickier for them was somewhat overwhelming for the students and they would get stumped on both so they felt like they couldn't move forward.

Several teams focussed on students' lack of background or comfort in coding in particular, and how it can create challenges for teachers. When reflecting on challenges, one team spoke about "trying to explain the code to students who are not very familiar with Scratch," and another about "trying to overcome students' hesitations about coding."

The final two challenges in Table 1 – *handling technological issues* and *working with classroom constraints* – relate to the resources teachers were using, which can pose an extra layer of difficulty. No comments were specific to coding. One of the teams working virtually mentioned encountering Internet issues and the difficulty of not being able to see students. Another team, working in-person, spoke about the challenge of using computers with all students sitting at desks facing forward:

> [The] orientation of the classroom was not optimal for computers because we couldn't see what is occurring on every computer screen, whereas if it was a circle the educators could keep better track.

4.1.3. Recommendations

One main recommendation that arose during the collective reflection session was, as one team put it: "Let students work together." One team highlighted both "group discussions" and "peer collaboration"; a second specified to "have students collaborate with others while building their codes, whether they work on it together or they ask questions to peers"; and a third suggested that students can "help one another out when they get stumped by explaining the code." Among the questionnaire reflections, one inservice teacher called "the amount of trouble shooting the students were doing for one another" a "huge success"; and another observed that "the students work well together and learn from each other, and this allows them to be successful." Collectively, it seemed the teachers were convinced that it was productive to *allow students to collaborate* while implementing coding in mathematics classrooms. As an additional component to the recommendation, some teams indicated a need to carefully pair students (e.g., "based on level of coding experience") and to teach students how to give their peers appropriate feedback (e.g., "encouraging students to explain what they did, not just give answers to peers").

Other productive strategies emerging from the collective reflection session aim to *ensure students have sufficient knowledge and/or scaffolding.* Many teams stressed the importance of students having the necessary background in coding concepts and languages. "For example, if the activity involves loops, review loops beforehand," one team recommended. In relation to activities involving Scratch, another team indicated that teachers should "front-load basic knowledge of Scratch (e.g., block colours, functions, etc.)." Two other teams making similar recommendations claimed that this can "allow students to explore comfortably" and that it may even be "essential" to the activity. A couple teams proposed that students should understand the mathematics needed for the activity first. One team, for instance, recommended "ensuring students are familiar with the math concepts and making the thinking explicit before implementing the coding aspect so it's a bit easier to code it." For other teams, the most important thing was that there was a separate teaching of mathematics and coding before they were combined. For example, one team suggested

> having a lesson for Python before and then integrating the math and Python together. Some [students] got overwhelmed or didn't understand how they were combining together. Do one thing at a time. Just Python, then just math, then integration.

Other possible scaffolding strategies were also mentioned: e.g., giving students some of the code or following a "use-modify-create" approach (where students first use given code, then modify it, and finally create their own). More specifically, one team recommended

beginning with a use and modify approach of the activity to familiarize students with programming while also focusing on math concepts before transitioning to the create stage.

The next six recommendations in Table 1 offer many more ideas to think about during the preparation of integrated coding and mathematics lessons. Based on their experiences, teams recommended that other teachers *do not limit coding tools to the computer* (e.g., "using white board, paper, pencil and other tools available in the class"), "*highlight the conventions of coding*," and *give sufficient time for student exploration* (e.g., "give students more time to explore (trial and error)"). Several teams also pointed to the idea of including ways to *check-in on students*: to survey their "progress," to "catch errors," and to incorporate "teaching moments." One team suggested including polls "about how students are feeling (not only what they are learning). [So] students can see how other students are feeling too." Another team proposed "having a worksheet where students can fill in the blanks to consolidate learning and work at their own pace." In connection with this is the recommendation to *create lessons that meet varying levels of need*. For example, some teams emphasized their productive use of "extension" questions, "as sometimes those who were familiar with Scratch completed the activity very quickly but had extension questions to further their learning." Another important idea was to *foster an environment that is accepting of errors and questions*; as one team explained, the classroom should be a place where "bugs are OK and the students don't have to know 100% what they are doing." One team recommended specifically "having a pep-talk on failure being normal in coding"; and then "when bugs/errors occur, have students develop a mindset to debug, not saying 'what did I do wrong' [but] 'what is happening here?'"

Although the teachers attributed an importance to being prepared (e.g., "work through the code yourself, prior to the lesson"), they also recognized the need to *just go with it* (especially in the case of coding) and to *acknowledge and accept that teachers are not experts*. One team explained: "There is more than one way to learn coding. If a lesson goes in a different direction, you may not want to shut it down … IF learning is occurring, let them have freedom." Two other teams told teachers to "expect the unexpected ways that students might find for solving problems" and that "sometimes they can come up with things you didn't know and surprise you with what they are able to come up with." This last comment suggests that teachers may be able to learn with and from their students. Indeed, one team explicitly recommended that teachers "learn along with the students," "experiment with them," and engage in "co-discovery," claiming that "to teach coding, you don't need to be an expert." Another team added: "It's okay to not be an expert and you have to accept you'll be learning alongside the students." Nevertheless, some teams also mentioned that, in general, teachers are in need of more professional development opportunities: "People in this project are willing to try the coding; how do you reach the 90% who cringe at the word 'code'?"

A final recommendation, related to the broader aims of the new curricular expectations and corresponding professional development, is to *make coding a consistent part of the classroom*: that is, to implement meaningful coding lessons on a continuous basis (e.g., "daily" or "once a week"). As one team put it: "Don't make it an EVENT, but an ongoing process."

4.2. Teachers' Reflections on the Collaborative Approach

Table 2 summarizes benefits, challenges, and recommendations that were recognized by preservice and inservice teacher teams in relation to the collaborative approach taken to implement coding in mathematics classrooms. These are further described, with the support of illustrative quotes, throughout Sections 4.2.1 (benefits) and 4.2.2 (challenge and recommendation). For an example of how one teacher team experienced some of these, see Sardella et al. (2023).

Table 2
Benefits, challenges, and recommendations related to the collaborative approach, according to participating teachers

Benefits
• Combining different expertise to lead to enriched, mutually beneficial experiences • Having new people and energy in the classroom • Bringing new ideas and approaches to the classroom • Providing additional role models in the classroom • Having someone with more experience in coding lead the lesson For inservice teachers: • Being free to walk around the room • Gaining familiarity with and confidence in coding • Seeing how coding can connect to mathematics • Seeing how to implement coding in mathematics classrooms • Learning new pedagogical approaches For preservice teachers: • Gaining experience in a classroom • Seeing "theory in action"
Challenge
• Making time for preparation
Recommendation
• Provide more time for preparation

4.2.1. Benefits

Teachers' reflections highlighted the value of the collaboration for all involved. In general, there was a sense that the approach allowed for *combining different expertise to lead to enriched, mutually beneficial experiences*. Preservice teachers often contributed their expertise in coding for mathematics (education),[12] while inservice teachers contributed their expertise in the classroom. One teacher's comment emphasized how the different parties were all learning from each other:

> Watching the [preservice teachers] interact with the kids, it was … of a different dynamic for sure … and [we were] learning from them as much as they were learning from the kids, and they were learning from us.

Some of these learning opportunities were specified by teachers, as elaborated below.

The newness and freshness that preservice teachers can offer through the collaboration is represented in several benefits, including *having new people and energy in the classroom*, *bringing new ideas and approaches to the classroom*, and *providing additional role models in the classroom*. When reflecting on having the preservice teachers in their class, one inservice teacher noted that one "thing that they brought was their enthusiasm" and "the kids themselves were right into it"; "it was just so refreshing." Another inservice teacher indicated that they "could never have gotten this far in coding without [the preservice teachers]" because they "didn't have that knowledge"; a third said similarly that the "fresh perspectives" from [the preservice teachers] and "their confidence in the subject matter put both the teacher and students at ease with the new materials." One team synthesized that the collaboration led to "doing things that [inservice] teachers and students didn't think of." Some inservice teachers also saw female preservice teachers as being able to "open up some eyes" specifically in relation to women in coding ("a largely male-dominated field").

Having someone with more experience in coding lead the lesson (the preservice teachers in our case) was noted as beneficial by some inservice teachers, since the leader was able to bring a better understanding and make students more comfortable. One teacher observed that the preservice teachers "could anticipate some of the problems the kids would have." Another teacher said it was

[12] See Section 3 for a description of preservice teachers' background in coding for mathematics.

"great" having "someone leading the lesson who had a deeper understanding of coding and specifically about Scratch" since their "own coding experience is quite limited." With the preservice teachers providing additional support and often taking the lead, this also meant that inservice teachers could experience the potential benefit of *being free to walk around the room* and interact with more students. One inservice teacher explained that they are typically "drawn to specific kids that need [their] help"; with the preservice teachers in the room, these students "got a new face," while the inservice teachers "got to go and see the other kids and their thought processes."

Inservice teachers mentioned several potential gains that were specific to them. As expected, many teachers' comments pointed to them *gaining familiarity with and confidence in coding*. Some teachers spoke generally, saying that they "learned so much more and about how Scratch and coding works" or that they now have "more background knowledge and experience in coding." Others were more specific about what they learned, speaking about "blocks and how they can run concurrently," "conditions within coding," "'if' and 'then' statements," or how they "feel much more comfortable with debugging and breaking the steps down into the small steps needed for coding." The increased comfort mentioned in the last comment may correlate with increased confidence, as the same teacher indicated that they have become more of a "risk-taker with coding." Another teacher said: "Going forward I have some confidence" with respect to implementing similar activities incorporating math concepts with programming. One teacher also mentioned that they "gained more insight into the importance of teaching coding to the students," highlighting that it "is a skill that will be very valuable for their future endeavors for employment."

Inservice teachers' comments also suggested that they may have experienced several other benefits related to gaining knowledge for implementing coding in mathematics classrooms. For instance, some comments related to *seeing how coding can connect to mathematics*; in particular, the mathematics the teachers are expected to teach, including how to "integrate coding with math concepts from different strands." One teacher spoke about coding's potential use in "spatial sense," while others saw a connection between "financial literacy" and using coding to "solve mathematical problems." More generally, some teachers said that they observed "the direct connection between math concepts" and the "coding commands." Several teachers also touched upon the benefit of *seeing how to implement coding in mathematics classrooms*; that is, gaining "insight into how to run a coding lesson." Some teachers mentioned specific insights, such as "understanding that pre-teaching would be essential" or that it could be helpful "having [students] explore conditionals with an easier problem." By working with preservice teachers who were learning about recent pedagogical approaches in their courses, there was also the possibility of inservice teachers *learning new pedagogical approaches*. For instance, one inservice teacher explained how an interaction with a preservice teacher opened their mind to a more student-centred experiential learning approach:

> It gave me a different perspective … traditionally we would put up something and then take it up [e.g., the teacher puts up a problem and then goes through a step-by-step solution] … then [the preservice teacher] says "well, don't take it up: have [the students] do this, and then build on this skill, and then articulate the whole process" … I had never thought of it that way.

In the collective reflection session, benefits relating specifically to preservice teachers also emerged. For example, the benefit of *gaining experience in a classroom* – "being in that learning environment and getting a feel for how it is teaching a class" – was highlighted. This includes experience specific to teaching coding, "something that might be in future teaching." One team mentioned how they got to see "how schools are integrating coding in the math curriculum" in their classes and in other ways (e.g., through "competitions with different schools"). The experience gained by preservice teachers may also be related to teaching more generally: e.g., experiencing the "unpredictable" nature of teaching a lesson and how teachers "must always be ready to adapt on the fly." Some comments also alluded to preservice teachers *seeing "theory in*

action": that is, seeing what they had learned in their courses within their teaching in schools, including affordances of coding for mathematics learning (e.g., access and agency; Gadanidis et al., 2017) and approaches to integrating coding in classrooms (e.g., the Use-Modify-Create model; Waite & Grover, 2020).

4.2.2. Challenge and recommendation

In regard to challenges related to the collaborative approach, pre/inservice teacher teams mentioned one particular aspect: *making time for preparation*. The collaboration required time for preservice and inservice teachers to plan, create, and discuss their lessons; to get students caught up with respect to coding and mathematics prior to preservice teachers entering the classroom; and to enable preservice teachers to get to know the students. When reflecting on challenges, one teacher team simply said: "Time (1 Meeting, 1 Planning Session, and we are LIVE)." Several inservice teachers mentioned that they "needed to do several pre lessons to get students ready." In relation to mathematics, one teacher indicated "fitting in the pre-learning of the spatial sense concepts thoroughly prior to the coding sessions." Other teachers explained, in relation to coding, that "in intermediate [classes] we needed to cover the majority of skills that should have been presented in grades K–6, as the exposure of coding … varied." It was also pointed out that students did not get the chance to meet the preservice teachers prior to the implementations. Some preservice teachers highlighted the difficulties associated with this: e.g., "We didn't really know what kind of students we'd be teaching, so we didn't know what to expect." Others also mentioned "knowing how comfortable students were with coding" or "understanding where students are" as challenges encountered during their participation in the collaboration.

Perhaps naturally, the recommendation we identified in teachers' reflections concerning the collaborative approach directly addresses the above challenge: namely, teachers recommended to *provide more time for preparation*. This includes "more time for planning between classroom teacher and [pre-service teachers]," as well as ensuring that preservice teachers give the inservice teacher a "run through" of the coding portions of the lesson to further support their understanding.

5. An Evaluation of our Collaborative Approach Using Teachers' Reflections

We see both parts of Section 4 as supporting an evaluation of the collaborative approach described in Section 3.

Section 4.1 offers a window into the kinds of concrete insights that teachers may gain through participating in the approach. The benefits of implementing coding in mathematics classrooms highlighted by preservice and inservice teachers (Section 4.1.1) are not necessarily surprising. Many of them were already proposed in the 70s and 80s with the pioneering work by Seymour Papert (e.g.,1980), who envisioned coding as a way of inviting all children to gain mastery over a powerful tool for doing mathematics as mathematicians do, including engaging in personally meaningful self-directed projects and learning related dispositions and processes. Some of the benefits have also been described in subsequent mathematics education research literature (e.g., Gadanidis et al., 2017) and reported by other teachers teaching coding in K–12 settings (see, e.g., Rich et al., 2019). What is promising is that the collaborative approach we took seemed to enable preservice and inservice teachers to experience these benefits together, in real-time. Such an experience may serve to motivate teachers, encouraging and framing their future implementations of coding in mathematics classrooms.

The challenges to implementation experienced by pre/inservice teacher teams (Section 4.1.2) are not necessarily surprising either, with some relating to the challenge of teaching in general (e.g., responding to students' needs) and others (e.g., ensuring students stay engaged and on task, addressing students' differences, supporting students who lack experience in coding, handling technological issues) being reported by other teachers adapting to new curricular demands involving coding (Rich et al., 2019; Sentance & Csizmadia, 2017; Vinnervik, 2022b). Within teachers' reflections on recommendations (Section 4.1.3), we can identify many productive

pedagogical strategies, which teachers may be able to use to overcome some of the identified challenges: for instance, creating lessons that meet varying levels of need and include sufficient scaffolding may assist in addressing students' differences and supporting students who lack experience in coding. Such recommendations, emerging from teachers' experiences, also reflect a basic understanding of ideas presented in emerging literature on strategies for teaching coding in K–12 (e.g., Grover, 2020, which includes chapters on universal design pedagogies and scaffolding approaches). Once again, this suggests a promising outlook on the collaborative approach we took.

Section 4.2 provides a more direct evaluation of the collaborative approach in terms of potential benefits and challenges. We could be critical of the claimed "benefit" of having someone with more coding experience (the preservice teachers in our case) lead the lesson, since such a reliance on others may promote one-time events (Vinnervik, 2022b) or limit the inservice teachers' growth (Kelter et al., 2021). However, other claimed benefits suggest that inservice teachers may have gained new pedagogical approaches, as well as familiarity with and confidence in coding, how it can connect to mathematics, and how it can be implemented in mathematics classrooms. We hypothesize that this was supported by the inservice teachers' interactions with and observations of the preservice teachers, who had recent experience with novel pedagogical approaches in education courses and a greater expertise in coding mathematics. The gains by inservice teachers may suggest that the collaborative approach is productive towards assisting in addressing a more "primary" challenge: that is, teachers having sufficient knowledge of programming to be able to integrate it in their subject areas (Grover & Yadav, 2020). Indeed, among the challenges identified by 313 primary teachers teaching programming, the most cited were personal in nature, with teachers' greatest concern being their own knowledge about programming (Rich et al., 2019). The study by Vinnervik (2022b) involving Gr. 1–9 mathematics teacher leaders who would soon be expected to integrate and support the integration of programming also found a sense of inadequate preparation among the teachers, with concerns being not only about their knowledge of programming, but also about their confidence in programming and a lack of teaching approaches that go beyond fun one-time events. Looking at the benefits in Table 2, the collaborative approach seemed to meet some needs identified by inservice teachers. Importantly, it also seemed to meet some needs of preservice teachers: for example, by providing them with authentic teaching experiences in the classroom (Mason & Rich, 2019; Suters, 2021), during which they can see what they have learned in action and gain pedagogical guidance from the more experienced inservice teachers.

We finish this section by highlighting one recommendation in Table 1: acknowledge and accept that teachers are not experts. While participating in our collaborative approach, some teachers realized that they do not need to have expert-level knowledge to implement coding in their classrooms; rather, they can experience coding activities alongside their students. We see this as particularly promising since it implies teachers' recognition that they can "just get started." Interestingly, this was found to be the most common advice given by the coding teachers surveyed by Rich et al. (2019), wherein a large group specified that teachers need to be willing to learn with and from their students:

> This role reversal has interesting implications for teaching and supports the imperative to 'just start.' As opposed to other well-known school subjects, teachers are suggesting that it's ok to not possess all the knowledge they want their students to gain and instead become the students themselves with their own pupils becoming the teachers (or at least co-learners). (p. 327)

6. Concluding Remarks

In this paper, we report on one possible approach for providing training simultaneously to preservice and inservice school teachers in response to new curricular demands to teach coding and integrate it in other school subjects such as mathematics. The approach involved grouping preservice and inservice teachers to plan and implement integrated coding and mathematics activities with school students, and then inviting those teachers to engage in a collective reflection

session to share insights they had gained through their implementations. Our own reflections on these insights suggest that the approach could be a promising way to support the enactment of new coding-related curricula, which could be of interest to other mathematics educators. In Sardella et al. (2023), we provide a complementary view of the collaborative approach through an example of one pre/inservice teacher team experience.

With new CT-infused curricula appearing, issues related to teacher training are particularly urgent, especially in light of past digitally driven reforms failing due to a lack of responsiveness to teachers' professional development needs (Vinnervik, 2022b). This paper adds to the growing collection of case studies reporting on possible effective approaches to subject-specific teacher training to address new curricular demands involving coding. Our case study aligns with other recent ones (e.g., Kelter et al., 2021; Suters, 2021), which acknowledge the importance of providing teachers with opportunities to practice teaching coding in authentic contexts, argued to be a necessary condition for supporting lasting changes in their practices (Mason & Rich, 2019). In contrast with previous studies, which typically focus on either inservice or preservice teachers, we focus on inservice and preservice teachers joining forces and learning from each other by bringing their respective experiences and skills to enact and reflect on new coding expectations in mathematics classrooms.

It is important to note the particularities of the preservice and inservice teachers participating in our collaborative approach. Having volunteered to participate, the inservice teachers were likely to have entered the experience with a self-identified need for training and positive dispositions towards the collaboration. We also hypothesize that they may have already bought-in to the new curriculum expectations and held positive attitudes towards the integration of coding in mathematics. As for the preservice teachers, they were required to participate as part of their workload in a mandatory upper-year mathematics course for future mathematics teachers. What may be more particular is that they all had a certain level of expertise and confidence in coding for mathematics, having learned how to use coding as a tool for investigating university-level mathematics since their first year of university. We expect that this influenced the collaborations, for example, enhancing the bidirectional gains by having preservice teachers that could act as more expert coders in the classroom.

Although preservice and inservice teachers claimed to gain pedagogical experience and knowledge through their collaborations, we do not know if this will lead to positive changes in their teaching practice or in their students' learning that also reflect the intentions of the curriculum (e.g., learning coding as a tool for solving mathematics problems). This aligns with much of the past work on professional development for teaching coding, which has not yet analyzed long-term and student-related impacts of training (Mason & Rich, 2019). A more careful analysis of transfer of learning and alignment with curricular expectations for both inservice and preservice teachers could be an interesting direction for future work and could provide a more detailed view of the impact of the training approach we reported. We also acknowledge that this is one possible approach and agree with Mason and Rich (2019) that there is a need for long-term, ongoing training for teachers who will be expected to teach coding (in an integrated manner). In our own research context, preservice teachers engage in integrated coding and mathematics activities over a sequence of university mathematics courses (the MICA courses), which span several years. Building on our past work documenting MICA students' development of coding as a personal investigation tool (e.g., Buteau et al., 2019; Buteau et al., 2020; Gueudet et al., 2022), we have started examining how the MICA courses may also support preservice teachers in developing coding as a pedagogical resource that will support their future work in schools (e.g., Broley et al., 2023; Sacristán et al., 2023).

We are looking forward to the next time we can realize the collaboration we described. In reflecting on our experience, we are noting ways in which we may productively adapt the training, with the aim of providing a richer experience for the participating teachers and school students: for example, by including preliminary workshops for inservice teachers to boost their confidence

in coding and awareness of novel pedagogical approaches; by expanding the time spent by participant teachers conducting integrated coding and mathematics activities in classrooms; by restricting participants to selecting activities from among good resources to relieve the stress of ensuring all participants proceed with quality activities; by encouraging groups of inservice teachers from different grades (possibly across elementary and secondary levels) to participate together and allow participants to experience how activities can build on one another from year to year; etc. We are also hoping that teachers who participated previously may serve as ambassadors for the approach to assist in getting new teachers involved and contribute to building computational modeling communities of practice in their schools.

Acknowledgements: We thank Carolyn Finlayson for her support with the preparation of preservice teachers to collaborate with the local teachers and schools. We also thank the Niagara Catholic District School Board, in particular numeracy consultants Laura Cronshaw and Jeff Martin, for their trust and making this initiative possible. Finally, we thank all of the preservice and inservice teacher participants, and the school students.

Author contributions: All authors are agreed with the results and conclusions.

Declaration of interest: None of the authors have a conflict of interest in this project.

Ethics declaration: Our submission does not require an Ethics Committee Approval because we are not reporting on a research study that involved human subjects.

Funding: The initiative reported in this paper was funded by the Mathematics Knowledge Network (Ontario, Canada). The work was also funded by the Social Sciences and Humanities Research Council of Canada (#435-2017-0367) and the Office of Experience Education at Brock University.

References

Araujo, R. C., Floyd, L., & Gadanidis, G. (2019). Teacher candidates' key understandings about computational thinking in mathematics and science education. *Journal of Computers in Mathematics and Science Teaching, 38*(3), 205–229.

Armoni, M. (2016). Computer science, computational thinking, programming, coding: The anomalies of transitivity in K-12 computer science education. *ACM Inroads, 7*(4), 24–27. https://doi.org/10.1145/3011071

Barcelos, T. S., Munoz, R., Villarroel, R., Merino, E., & Silveira, I. (2018). Mathematics learning through computational thinking: A systematic literature review. *Journal of University Computer Science, 24*(7), 815–845. https://doi.org/10.14210/jcthink.v2.n1.p23

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. *Digital Experiences in Mathematics Education, 3*, 115-138. https://doi.org/10.1007/s40751-017-0028-x

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education: Implications for policy and practice (JRC Science for Policy Report)*. European Commission. https://doi.org/10.2791/792158

Bocconi, S., Chioccariello, A., & Earp, J. (2018). *The Nordic approach to introducing computational thinking and programming in compulsory education*. Nordic@BETT2018 Steering Group. https://www.itd.cnr.it/doc/CompuThinkNordic.pdf

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the Annual Meeting of the American Educational Research Association* (pp. 1–25).

Broley, L., Buteau, C., & Muller, E. (2023). Coding in math learning: A 'triple instrumental genesis' approach to support the transition from university learner to school teacher. In M. Trigueros, B. Barquero, R. Hochmuth & J. Peters (Eds.), *Proceedings of the Fourth Conference of the International Network for Didactic Research in University Mathematics* (INDRUM 2022, 19-22 October 2022). Hannover, University of Hannover and INDRUM.

Buteau, C., Gueudet, G., Muller, E., Mgombelo, J., & Sacristán, A. I. (2019). University students turning computer programming into an instrument for "authentic" mathematical work. *International Journal of Mathematical Education in Science and Technology, 57*(7), 1020–1041. https://doi.org/10.1080/0020739X.2019.1648892

Buteau, C., Muller, E, Marshall, N, Sacristán, A.I., & Mgombelo, J. (2016). Undergraduate mathematics students appropriating programming as a tool for modelling, simulation, and visualization: A case study. *Digital Experience in Mathematics Education, 2*, 142–166. https://doi.org/10.1007/s40751-016-0017-5

Buteau, C., Muller, E., Mgombelo, J., Sacristán, A., & Driese, K. (2020). Instrumental genesis stages of programming for mathematical work. *Digital Experience in Mathematics Education, 6*(3), 367–390. https://doi.org/10.1007/s40751-020-00060-w

Buteau, C., Muller, E., & Ralph, B. (2015). Integration of programming in the undergraduate mathematics program at Brock University. In *Online proceedings of the Math + Coding Symposium*.

Caspersen, M.E. (2018) Teaching programming. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer Science Education: Perspectives on Teaching and Learning in School* (pp. 109–130). Bloomsbury Academic.

Cuny, J., Snyder, L., & Wing, J. M. (2010). *Demystifying computational thinking for non-computer scientists* [Unpublished manuscript, referenced in Wing, 2010].

Dagienė, V., Jevsikova, T., & Stupurienė, G. (2019). Introducing informatics in primary education: Curriculum and teachers' perspectives. In S. Pozdniakov, & V. Dagienė (Eds.), *ISSEP 2019: Informatics in Schools. New Ideas in School Informatics. Lecture Notes in Computer Science* (pp. 83–94), vol 11913. Springer. https://doi.org/10.1007/978-3-030-33759-9_7

Floyd, S. (2022). *The past, present, and future direction of computer science curriculum in K-12 education* [Doctoral dissertation, Western University]. Electronic Thesis and Dissertation Repository. https://ir.lib.uwo.ca/etd/8463/

Gadanidis, G., Cendros, R., Floyd, L., & Namukasa, I. (2017). Computational thinking in mathematics teacher education. *Contemporary Issues in Technology and Teacher Education, 17*(4), 458–477.

Gadanidis, G., Hughes, J.M., Minniti, L., & White, B.J.G. (2017). Computational thinking, Grade 1 students and the Binomial Theorem. *Digital Experiences in Mathematics Education, 3*, 77–96. https://doi.org/10.1007/s40751-016-0019-3

Gannon, S., & Buteau, C. (2018). Integration of computational thinking in Canadian provinces. In *Online Proceedings of the Computational Thinking in Mathematics Education Symposium*. http://ctmath.ca/wp-content/uploads/2018/10/Symposium_CanadaMap_Gannon-Buteau.pdf

Grover, S. (Ed.). (2020). *Computer science in K-12: An A to Z handbook on teaching programming*. Edfinity.

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. In S. Sentence, E. Barendsen, & S Carsten (Eds.), *Computer science education: Perspectives on teaching and learning in school* (pp. 19–38). Bloomsbury.

Grover, S., & Yadav, A. (2020). Integrating programming into other subjects. In S. Grover (Ed.), *Computer science in K-12: An A to Z handbook on teaching programming* (pp. 83–98). Edfinity.

Gueudet, G., Bueno-Ravel, L., Modeste, S., & Trouche, L. (2017). Curriculum in France. A national frame in transition. In D. R. Thompson, M. A. Huntley, C. Suurtamm (Eds.), *International Perspectives on Mathematics Curriculum* (pp. 41–70). Research Issues in Mathematics Education series, IAP.

Gueudet, G., Buteau, C., Muller, E., Mogombelo, J., Sacristán, A. I., & Santacruz Rodriguez, M. (2022). Development and evolution of instrumented schemes: a case study of learning programming for mathematical investigations. *Educational Studies in Mathematics, 110*, 353–377. https://doi.org/10.1007/s10649-021-10133-1

Guzdial, M. (2019). Computing for other disciplines. In S. Fincher, & A. Robins (Eds.), *The Cambridge Handbook of Computing Education Research* (pp. 584–605). Cambridge Handbooks in Psychology.

Kelter, J., Peel, A., Bain, C., Anton, G., Dabholkar, S., Horn, M. S., & Wilensky, U. (2021). Constructionist co-design: A dual approach to curriculum and professional development. *British Journal of Educational Technology, 52*(3), 1043–1059. https://doi.org/10.1111/bjet.13084

Ketelhut, D. J., Mills, K., Hestness, E., Cabrera, L., Plane, J., & McGinnis, J. R. (2020). Teacher change following a professional development experience in integrating computational thinking into elementary science. *Journal of Science Education and Technology, 29*(1), 174–188. https://doi.org/10.1007/s10956-019-09798-4

Mason, S. L., & Rich, P. J. (2019). Preparing elementary school teachers to teach computing, coding, and computational thinking. *Contemporary Issues in Technology and Teacher Education, 19*(4), 790–824.

Misfeldt, M., Szabo, A., & Helenius, O. (2019). Surveying teachers' conception of programming as a

mathematics topic following the implementation of a new mathematics curriculum. In U.T. Jankvist, M. van den Heuvel-Panhuizen, & M. Veldhuis (Eds.), *Proceedings of the Eleventh Congress of the European Society for Research in Mathematics Education* (pp. 2713–2720). Freudenthal Group & Freudenthal Institute, Utrecht University, Netherlands, and ERME.

Ontario Ministry of Education. (2020). *Ontario curriculum and resources: Elementary mathematics*. https://www.dcp.edu.gov.on.ca/en/curriculum/elementary-mathematics

Ontario Ministry of Education. (2021). *Ontario curriculum and resources: Grade 9 mathematics*. https://www.dcp.edu.gov.on.ca/en/curriculum/secondary-mathematics/courses/mth1w

Ontario Ministry of Education. (2022). *Ontario curriculum and resources: Elementary science and technology*. https://www.dcp.edu.gov.on.ca/en/curriculum/science-technology

Organisation for Economic Cooperation and Development (OECD). (2018). *PISA 2021 mathematics framework (draft)*. https://www.oecd.org/pisa/sitedocument/PISA-2021-mathematics-framework.pdf

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.

Reichert, J. T., Barone, D. A. C., & Kist, M. (2020). Computational thinking in K-12: An analysis with mathematics teachers. *EURASIA Journal of Mathematics, Science and Technology Education, 16*(6), https://doi.org/10.29333/ejmste/7832

Rich, P.J., Browning, S.F., Perkins, M., Shoop, T., Yoshikawa, E., & Belikov, O.M. (2019). Coding in K-8: International trends in teaching elementary/primary computing. *TechTrends, 63,* 311–329. https://doi.org/10.1007/s11528-018-0295-4

Sacristán, A.I., Santacruz-R., M., Buteau, C., Mgombelo, J., & Muller, E. (2023). Future teachers' appropriation of computer programming as a mathematical instrument and a resource for teaching. In H.-G. Weigand, A. Donevska-Todorova, E. Faggiano, P. Iannone, J. Medová, et al. (Eds), *Proceedings of the 13th ERME Topic Conference (ETC13) on Mathematics Education in Digital Age* (pp. 256-263). Constantine the Philosopher University in Nitra.

Sardella, J., Broley, L., & Buteau, C. (2023). Code mountain: Climbing it collaboratively. *Ontario Mathematics Gazette, 61*(3), 42–46.

Sentance S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies, 22*, 469–495. https://doi.org/10.1007/s10639-016-9482-0

Suters, L. (2021). Elementary preservice teacher coursework design for developing science and mathematics computational thinking practices. *Contemporary Issues in Technology and Teacher Education, 21*(2), 360–440.

Tissenbaum, M., Weintrop, D., Holbert, N., & Clegg, T. (2021). The case for alternative endpoints in computing education. *British Journal of Educational Technology, 52*(3), 1164–1177. https://doi.org/10.1111/bjet.13072

Vinnervik, P. (2022a). An in-depth analysis of programming in the Swedish school curriculum—rationale, knowledge content and teacher guidance. *Journal of Computers in Education*, 1–35. https://doi.org/10.1007/s40692-022-00230-2

Vinnervik, P. (2022b). Implementing programming in school mathematics and technology: Teachers intrinsic and extrinsic challenges. *International Journal of Technology and Design Education, 32*, 213–242. https://doi.org/10.1007/s10798-020-09602-0

Waite, J., & Grover, S. (2020). Worked examples & other scaffolding strategies. In S. Grover (Ed.), *Computer Science in K–12: An A to Z Handbook on Teaching Programming* (pp. 240–249). Edfinity.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*, 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2010). *Computational thinking: What and why?* [Unpublished manuscript]. https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Wu, L., Looi, C.K., Multisilta, J., How, M.-L., Choi, H., Hsu, T.-C., & Tuomi, P. (2020). Teacher's perceptions and readiness to teach coding skills: A comparative study between Finland, Mainland China, Singapore, Taiwan, and South Korea. *The Asia-Pacific Education Researcher, 29*, 21–34. https://doi.org/10.1007/s40299-019-00485-x

Yadav, A., Mayfield, M., Zhou, N, Hambrusch, S., & Korb, J.T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education, 14*(1), 1–16. https://doi.org/10.1145/2576872