This framework for developing pre-service teachers' knowledge does not necessarily depend on computers or other educational technology.

BY AMAN YADAV, CHRIS STEPHENSON, AND HAI HONG

Computational Thinking for Teacher Education

ENTHUSIASM HAS GROWN in recent years for computer science education in many countries, including Australia, the U.S, and the U.K.^{14,15} For example, in 2012, the Royal Society in the U.K. said, “Every child should have the opportunity to learn concepts and principles from computing, including computer

science and information technology, from the beginning of primary education onward, and by age 14 should be able to choose to study toward a recognized qualification in these areas.”²⁶ And in 2016, the College Board in the U.S. launched a new computer science curriculum for high schools called “Computer Science Principles”⁶ focusing on exposing students to computational thinking and practices to help them understand how computing influences the world. Within the computer science education community, computational thinking is a familiar term, but among K–12 teachers, administrators, and teacher educators there is confusion about what it entails.

Computational thinking is often mistakenly equated with using computer technology.^{11,29} In order to address this misrepresentation, the scope of this article includes a definition of computa-

» key insights

- Few teacher-education programs focus on training pre-service teachers to incorporate computational thinking into K–12 classrooms.
- Redesign of courses on educational technology and methods is critical to developing pre-service teacher competencies in computational thinking.
- Education and computer science faculty should work collaboratively, using their complementary expertise in computing and teacher development.

tional thinking and the core constructs that would make it relevant for key stakeholders from K–12 education and teacher-training programs.

Denning suggested¹³ that the idea of computational thinking has been present since the 1950s and 1960s “as algorithmic thinking,” referring specifically to using an ordered and precise sequence of steps to solve problems and (when appropriate) a computer to automate that process. Today, the term “computational thinking” is defined by Wing²⁸ as “solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science.” Computational thinking also involves “using abstraction and decomposition when attacking a large complex task or designing a large complex system.”²⁸ A report on computational thinking by the National Council for Research suggested it is a cognitive skill the average person is expected to possess. For example, the cognitive aspects of computational thinking involve the use of heuristics, a problem-solving approach that involves applying a general rule of thumb or strategy that may lead to a solution.²⁸ This heuristic process involves searching for strategies that generally produce the right solution but do not always guarantee a solution to the problem. For example, “asking for directions in an unfamiliar place” from a local usually leads one to the right place, but one could also end up at a wrong place, depending on one’s understanding of local geography.¹⁷ Heuristic processes can be contrasted

with algorithms that guarantee success if followed correctly.¹⁷

Computational thinking has been suggested as an analytical thinking skill that draws on concepts from computer science but is a fundamental skill used by, and useful for, all people.²⁸ Some have argued that computational thinking is a practice that is central to all sciences, not just computer science.^{13,28} Bundy,⁴ for example, noted that computational thinking concepts have been used in other disciplines through problem-solving processes and the ability to think computationally is essential to every discipline. These powerful ideas and processes have begun to have significant influence in multiple fields, including biology, journalism, finance, and archaeology,²² making it important to include computational thinking as a priority for K–12 education. Wing²⁸ said, “To reading, writing, and arithmetic, we should add computational thinking to every child’s analytical ability.” In summary, computational thinking is a set of problem-solving thought processes derived from computer science but applicable in any domain.

Embedding computational thinking in K–12 teaching and learning requires teacher educators to prepare teachers to support students’ understanding of computational thinking concepts and their application to the disciplinary knowledge of each subject area. Specifically, teacher educators need to provide teachers with the content, pedagogy, and instructional strategies needed to incorporate computational

thinking into their curricula and practice in meaningful ways, enabling their students to use its core concepts and dispositions to solve discipline-specific and interdisciplinary problems. It is important to acknowledge that the current lack of an agreed-upon, exclusive definition of the elements of computational thinking makes it a challenge to develop clear pathways for pre-service teachers to be educated teachers—computationally.³⁰ Nevertheless, it is both important and possible to begin taking steps in this direction.

Here, we argue that, given the cross-disciplinary nature of computational thinking and the need to address educational reforms—Next Generation Science Standards and Common Core—it is beneficial to prepare teachers to incorporate computational thinking concepts and practices into K–12 classrooms. While most current efforts to embed computational thinking focus on in-service professional development, we posit that pre-service teacher education is an opportune time to provide future teachers with the knowledge and understanding they require to successfully integrate computational thinking into their curricula and practice. The following sections discuss the relevance of computational thinking constructs in K–12 education. We also discuss how to embed computational thinking into classrooms by using it as a methodology for teaching programming. In addition, we provide examples of how teachers in various disciplines can use computational thinking to address and enhance

Edtech start-up pavilion at International Society for Technology in Education conference.

existing learning outcomes. Finally, we discuss ways to implement computational thinking into pre-service teacher training, including how teacher educators and computer science educators can collaborate to develop pathways to help pre-service teachers become computationally educated.

What Is Computational Thinking?

How do we define computational thinking and use the definition as a framework to embed it in K–12 classrooms? Wing’s seminal column²⁸ offered a promising definition of computational thinking: “... breaking down a difficult problem into more familiar ones that we can solve (problem decomposition), using a set of rules to find solutions (algorithms), and using abstractions to generalize those solutions to similar problems.” Finally, automation is the ultimate step in computational thinking that can be implemented through computing tools. These concepts cut across disciplines and could be embedded across subjects in elementary and secondary schools. Based on this definition, a steering committee formed by the Computer Science Teachers Association (CSTA <https://www.csteachers.org/>) and the International Society for Technology in Education (ISTE <https://www.iste.org/>) presented a computational thinking framework for K–12 schools in 2011 with nine core computational thinking concepts and capabilities, including data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, parallelization, and simulation. They were also discussed in 2015 in the Computing at School (CAS) framework and guide for teachers to enable teachers in the U.K. to incorporate computational thinking into their teaching work.¹⁰ CSTA/ISTE and CAS also provide pedagogical approaches to embed these capabilities across the curriculum in elementary and secondary classes. For example, CSTA/ISTE describes how the nine core computational thinking concepts and capabilities could be practiced in science classrooms by collecting and analyzing data from experiments (data collection and data analysis) and summarizing that data (data representation).

Computational thinking is often mistakenly equated with using computer technology.

Algorithms are central to both computer science and computational thinking. Algorithms underlie the most basic tasks everyone engages in, from following a simple cooking recipe to providing complicated driving directions. Because there is a general misconception that algorithms are used only to solve mathematical problems and are not applicable in other disciplines,²⁹ it is important to introduce students to algorithms by first using examples from their daily lives. For example, in early grades, teachers could highlight the steps involved in brushing teeth, while in later grades, students could engage in following steps during a lab experiment. Understanding algorithms as a set of precise steps provides the basis for understanding how to develop an algorithm that can be implemented in a computing program. Students can be exposed to the computational thinking concept of abstraction by creating models of physics entities (such as a model of the solar system).³ Abstraction helps students learn to strip away complexity in order to reduce an artifact to its essence and still be able to know what the artifact is. In another example, Barr and Stephenson³ suggested students can learn about parallelization by simultaneously running experiments with different parameters. A number of leading research, educational, and funding organizations have argued for the need to introduce K–12 students to these core constructs and practices.

Computational Thinking in K–12 Education

The National Research Council (NRC)²² highlighted the importance of exposing students to computational thinking notions early in their school years and helping them to understand when and how to apply these essential skills.^{3,22} The NRC report²² on the scope and nature of computational thinking highlighted the need for students to learn the related strategies from knowledgeable educators who model these strategies and guide their students to use them independently. Similarly, Barr and Stephenson³ argued that, given that students will go into a workforce heavily influenced by computing, it is important for them to begin to work with computational thinking ideas and

tools in grades K–12. Specifically, they discussed³ the need to highlight “algorithmic problem solving practices and applications of computing across disciplines, and help integrate the application of computational methods and tools across diverse areas of learning.”

Recent educational reform movements (such as the Next Generation Science Standards and the Common Core) have also focused on computational thinking as a key skill for K–12 students. For example, the Next Generation Science Standards (NGSS) have identified computational thinking as key scientific and engineering practices that must be understood and applied in learning about the sciences.²⁰ Computational theories, information technologies, and algorithms played a key role in science and engineering in the 20th century; hence, NGSS suggested allowing students to explore datasets using computational and mathematical tools. One example of embedding computational thinking in science classrooms is Project GUTS (Growing Up Thinking Scientifically), which highlights what computational thinking looks like for students using three domains: modeling and simulation, robotics, and game design.¹⁸ Using the StarLogo computational environment, Project GUTS focuses on abstraction, automation, and analysis through a use-modify-create learning progression to allow students to use the tools, as well as modify and create them, thus deepening their acquisition of computational thinking concepts in the context of science learning. Similarly, the Scalable Game Design project engages students in computational thinking concepts through game and simulation design in science classes.²³

While such efforts involve embedded computational thinking in elementary and secondary subject areas, the College Board, with support from the National Science Foundation, has led the effort to introduce students to computational thinking constructs through a standalone advanced-placement course called Computer Science Principles designed to go beyond programming and focus on computational thinking practices to “help students coordinate and make sense of knowledge to accomplish a goal or task.”⁶ The course includes six computational thinking

practices: connecting computing, creating computational artifacts, abstracting, analyzing problems and artifacts, communicating, and collaborating. They are designed to allow students to develop a deep understanding of computational content and how computing is changing our world.⁶ The course is also structured around seven big ideas: creativity, abstraction, data and information, algorithms, programming, the Internet, and global impact. These big ideas from computer science overlap with the computational thinking concepts detailed in the CSTA/ISTE framework outlined earlier, but the Computer Science Principles course also adds programming as a fundamental concept. Programming is the next step in the computational thinking framework, allowing students to develop software and create computational artifacts like visualizations.⁶

Programming allows students to develop and execute algorithms while providing opportunities for them to show creative expression, create new knowledge, and solve problems.⁶ While programming is one of big ideas of Computer Science Principles, the goal is to go beyond learning one particular type of programming language to how computing tools can be used to solve problems through an iterative process.⁶ The Computer Science Principles framework is being used by a number of leading educational organizations across the U.S. to instantiate different versions of the Computer Science Principles course. For example, Project Lead The Way, a nonprofit organization that provides a transformative learning experience for K–12 students and teachers across the U.S. through pathways in computer science, engineering, and biomedical science has rolled out its version of the Computer Science Principles course to more than 400 schools. Another organization, Code.org, is rolling out its own advanced placement computer science curriculum to public school districts across the U.S.

While embedding computational thinking in STEM subject areas or through standalone courses is an important effort, the trans-disciplinary nature of computational thinking competencies provides an opportunity to integrate computational thinking

ideas into all K–12 subject areas. The goal of computational thinking, said Hemmendinger,¹⁶ is “to teach them [students] how to think like an economist, a physicist, an artist, and to understand how to use computation to solve their problems, to create, and to discover new questions that can fruitfully be explored.” Research on embedding computational thinking in K–12 is also starting to emerge and has suggested that students exposed to computational thinking show significant improvement in their problem-solving and critical-thinking skills.⁵ A 2015 study by Calao et al.⁵ reported that integrating computational thinking in a sixth-grade mathematics class significantly improved students’ understanding of mathematics processes when compared to a control group that did not learn computational thinking in their math class.

Although computational thinking is deeply connected to the activity of programming, it is not essential to teach programming as part of a pre-service computational thinking course. Such courses should focus on computational thinking within the context of the teachers’ content areas. Those interested in programming should have access to standalone courses that focus more specifically on programming and computer science. Despite the current lack of clarity as to the definition of computational thinking, Wing’s ideas provide a good starting point for conceptualizing it for teacher educators. Similarly, the computational thinking concepts and practices outlined in the CSTA/ISTE framework exemplify how these concepts can be used across the curriculum and to prepare pre-service teachers. The rest of this article focuses on how to develop pre-service teacher computational thinking competencies not related to programming to allow them to teach computational thinking ideas in their classrooms.

To achieve these goals, we need to prepare new teachers who are able to incorporate computational thinking skills into their discipline and teaching practice so they can guide their students to use computational thinking strategies.²² The following section discusses how the education community can prepare teachers to embed computational thinking in their curricula and

practice and offers recommendations to prepare the next generation of computationally literate teachers.

Computational Thinking and Teacher Education

As discussed previously, researchers have argued that computational thinking needs to be on par with reading, writing, and arithmetic.^{3,28} Recent efforts to train teachers to embed computational thinking have focused on in-service teacher professional development,¹⁸ but there is limited understanding of how to engage pre-service teachers from other content areas in computer science and computational thinking.²⁹ This complication is compounded by the fact that few teacher-preparation institutions offer programs specifically for computer science teachers. Furthermore, certification and licensure of computer science teachers is deeply flawed, as detailed in the “Bugs in the System” report by the Computer Science Teacher Association.⁸ It is vital that teacher education programs address the lack of teacher training around computer science ideas, given the burgeoning grassroots movement and impetus from governments to expand computer science learning opportunities in elementary and secondary classrooms, including the Computer Science For All initiative launched January 2016 in the U.S.

So how do teacher educators develop mechanisms to expose pre-service teachers to computational thinking constructs and understanding within the context of their subject areas? How do we develop pre-service teachers’ knowledgebase so they can provide relevant, engaging, and meaningful computational thinking experiences for their students? Darling-Hammond and Bransford¹² proposed a framework that could be adapted to prepare teachers to incorporate computational thinking, articulating knowledge, skills, and dispositions that teachers should acquire and suggesting that teachers need knowledge of learners, as well as of subject matter and curriculum goals.

Teacher educators need to first develop pre-service teachers’ knowledge and skills on how to think computationally and then how to teach their students to think computationally. It is thus impera-

Project Lead the Way session at San José State University, San Jose, CA.

tive for pre-service teachers to understand computational thinking in the context of the subject area they will be teaching. This requires them to have deep understanding of their own discipline and knowledge of how computational thinking concepts relate to what students are learning in the classroom.²² Moreover, the NRC report on the pedagogical aspects of computational thinking argued that teaching this content could put teachers in new and unfamiliar roles in classrooms where students collaborate to solve complex problems. It is thus important that teacher educators “build on what teachers know and feel comfortable doing.”²¹

Developing pre-service teachers’ competencies to embed computational thinking in their future classrooms requires that they are taught to think computationally, as well as how to teach their students to think computationally, especially in the context of specific subject areas. Teacher-training programs are a natural place to introduce teachers to computational thinking and how to incorporate it in their content. A 2014 study by Yadav et al.²⁹ examined the influence of a one-week computational thinking module on pre-service teachers’ understanding and attitudes toward embedding computational thinking in their future classrooms. Pre-service teachers enrolled in a required introductory educational psychology course were divided into two groups. One (the con-

trol group) did not experience the computational thinking module, while the second (the experimental group) specifically learned about computational thinking ideas by working through the module. The authors found the majority of pre-service teachers in the control group viewed computational thinking as integration of technology in the classroom, whereas the majority of participants in the experimental group developed their understanding of computational thinking as a problem-solving approach by using algorithms/heuristics. Results also suggested pre-service teachers in the experimental group were better able to articulate how to integrate computational thinking in K–12 classrooms as compared to the control group. The results from the study indicate the potential to integrate computational thinking for pre-service teachers within existing teacher-education courses. The authors used examples from daily life, as well as discipline-specific examples, to highlight computational thinking to pre-service teachers. For example, they used an example of giving directions from point A to point B to highlight what an algorithm is (a step-by-step route), the efficiency of algorithms (how to provide the best way to get from A to B), abstraction (how to effectively give any direction), and automation (how to design a system like Google Maps). In another example, Yadav et al.²⁹ showcased the idea of parallel processing by discuss-

ing the quickest way for two friends to buy movie tickets when three lines are available; see the Computational Thinking Modules at http://cs4edu.cs.purdue.edu/comp_think for other examples of how computational thinking constructs were highlighted for pre-service teachers. However, the study by Yadav et al.²⁹ was conducted in a general teacher-education course for pre-service teachers from all content areas, next steps should involve embedding computational thinking concepts into courses for teachers of specific subject areas.

Given the strict sequence of courses for teacher education students, teacher educators need to expose pre-service teachers to computational thinking ideas and competencies through existing coursework. One opportunity that offers a natural fit is to introduce computational thinking within existing educational-technology courses in teacher-education programs. These

courses are typically disconnected from the teaching theories and methods pre-service teachers learn in other education courses, focusing instead on technology (such as Web 2.0 tools to teach).¹⁹ Rather than focus on using educational technology tools, educational-technology courses should be revised to provide pre-service teachers with opportunities to think computationally and experience computational thinking as a generic set of skills and competencies that do not necessarily depend on computers or other educational technology.

Redesigning introduction-to-educational-technology courses around learning core computational thinking concepts and capabilities is also an opportunity for computer science and education faculty to work together. Taylor²⁵ wrote that many early courses focus on simple programming intended to help students “learn something both

about how computers work and how his or her own thinking works.” About a decade ago, however, educational-technology courses moved away from this view and began to focus instead on use of predesigned software tools in the classroom. The recent burgeoning movement around computational thinking is an opportunity to reset and redesign educational technology courses, making them both more relevant and more rigorous.

Given the importance of exposing pre-service teachers to computational thinking in the context of their discipline, educational technology courses could be customized for groups of pre-service teachers based on their subject areas and tied to their day-to-day classroom activities. The Technological Pedagogical Content Knowledge (TPACK) framework¹⁹ is a useful model for integrating computational thinking where the related ideas are closely knit within the subject matter and pedagogical approaches pre-service teachers will teach in their future classrooms. TPACK extends Shulman’s idea²⁴ of pedagogical content knowledge by including knowledge teachers need to teach effectively with technology.¹⁹ TPACK suggested teachers learn about effective technology integration within the context of subject matter and pedagogy; similarly, teachers need to develop computational thinking knowledge within the context of their content knowledge and pedagogical knowledge.

Methods courses in teacher-education programs also provide an opportunity to help pre-service teachers incorporate computational thinking in the context of their future subject areas. Methods courses enable them to acquire new ways to think about teaching and learning in one particular subject area and provide opportunities for “developing pedagogical ways of doing, acting, and being as a teacher.”¹ A methods course weaves “together knowledge about subject matter with knowledge about children and how they learn, about the teacher’s role, and about classroom life and its role in student learning.”¹ Within this context, a methods course could also be a place where pre-service teachers explore computational thinking ideas within the context of their specific subject-

Developing pre-service teacher computational thinking.

Recommendations for computational thinking in teacher preparation.

Curriculum. Develop a pre-service teacher education curriculum to prepare teachers to embed computational thinking in their classrooms.

Core ideas. Introduce pre-service teachers to core ideas of computational thinking by redesigning educational technology courses.

Methods courses. Use elementary and secondary methods courses to develop pre-service teachers’ understanding of computational thinking in the context of the discipline.

Collaboration. Computer science educators and teacher educators collaborate on developing computational thinking curricula that goes beyond programming.

Teacher education. Use existing resources and curriculum standards to assimilate computational thinking into pre-service teacher education.

area specializations. For example, in a Methods of Teaching English course, prospective teachers could learn to embed algorithms into a writing activity by asking students to write a detailed recipe—a step-by-step series of instructions—for a favorite food. Similarly, pre-service teachers in a social studies methods course could learn to incorporate data analysis and pattern recognition by having students collect and analyze population statistics and use it to identify and represent trends.³ Data-analysis tools could be as simple as Piktochart, which allows students to represent data and information through infographics, to more advanced tools like Google Charts, which allow students to dynamically represent data using customizable and interactive charts. Pre-service teachers in a science-methods course could be exposed to computational thinking through computational models for demonstrating scientific ideas and phenomena to their future students.²⁷ The computational models could also be used to test hypotheses, as well as solutions to problems.²³ As discussed earlier in this article, students could use tools like StarLogo and Scalable Game Design to develop computational models as they engage in simulation and game design.

The general concepts of computational thinking acquired in the educational-technology course and the discipline-specific computational thinking practices acquired in the methods courses would help pre-service teachers connect computational thinking to content they will cover in their future classrooms. In this way, the constructs of pedagogical content knowledge²⁴ and technological pedagogical content knowledge¹⁹ provide support for developing pre-service teachers' computational thinking knowledge. Specifically, educational-technology courses would serve as a foundation for developing content knowledge for computational thinking. This knowledge would allow teachers to explore core computational thinking ideas, why those ideas are central, and how computational thinking constructs are similar to or differ from other parallel concepts (such as mathematical thinking).²⁴ As pre-service teachers take teaching-methods courses, they

would learn to integrate computational thinking into the context of particular subject areas. This would allow them to learn how to represent and formulate computational thinking in the subject and make it comprehensible to students.²⁴ By engaging pre-service teachers in computational thinking ideas in the context of teaching their content area, teacher educators could better ensure it becomes part of their own and their students' vocabulary and problem-solving tool set.

The Computational Thinking Progression Chart³ provides a starting framework around which teacher educators could begin to shape pre-service teacher experiences in elementary and secondary teacher-education programs. Within elementary education, incorporating computational thinking exercises into literacy learning offers a straightforward transition for teacher candidates. For example, pre-service teachers would be able to explore how to include abstraction into the analysis of themes within prose or poems using textual details or in summarizing text.⁷ They could also build a lesson plan that incorporates data analysis and data representation by having students identify words that depict feelings and comparing how they are represented across different versions of the same story. Similarly, pre-service teachers could embed computational thinking into lesson plans for language arts at the secondary level by allowing students to collect and integrate data/information from multiple sources to visually represent common themes. Elementary- and secondary-level pre-service science teachers could include data collection, analysis, and representation into any activity in which students gather data and identify and represent patterns in that data. Finally, social studies pre-service teachers could explore how to use large datasets (such as census data) to enable students to explore and identify patterns and discuss the implications of the increasing access to large amounts of personal data.

While existing teacher-education courses provide opportunities to introduce pre-service teachers to computational thinking, some programs might consider developing standalone courses and/or certificate programs that allow pre-service teachers to discover

the scope of computational thinking concepts and capabilities, as well as engage with computational tools that nurture development of computational thinking competencies. These courses would ideally be developed by education faculty and computer science faculty collaborating to identify appropriate learning outcomes and available resources.

Because integrating computational thinking into any curriculum involves exposing teachers and students to concepts and practices used by computer scientists, it is important for teacher educators to work closely with computer science faculty. Similarly, education faculty have a nuanced understanding of K–12 curriculum and educational policies that are key to ensuring current computational thinking efforts are successful. A 2016 report by the Computing Research Association⁹ highlighted the need for computer science faculty to establish interdisciplinary connections with colleagues from other disciplines (such as teacher education, educational psychology, and learning sciences). These collaborations included co-developing and co-teaching courses that prepare teachers to teach computational thinking; see Yadav and Korb³¹ for what such a course might look like. Furthermore, faculty could have joint appointments in education and computer science that would enable them to jointly develop programs and collaborate on research around teaching computational thinking.⁸ This would enable computer scientists and teacher educators to collaborate on developing both plugged and unplugged activities to expose pre-service teachers to computational thinking and its implementation. The accompanying figure showcases the interconnectedness of our recommendation for developing pre-service teacher competencies as computer science and teacher educators collaborate to develop computational thinking understanding through educational-technology courses. Pre-service teachers then learn how to use that knowledge to teach children to think computationally in the context of a particular subject area through methods courses.

Additionally, many available resources could be incorporated into an educational-technology or subject-spe-

cific methods course that could help pre-service teachers connect computational thinking to their daily lives and to classroom contexts. For example, pre-service teachers could carry out “CS Unplugged” activities (<http://csunplugged.org/>), many of which teach computational thinking skills without needing a computer and are easily adapted to other subjects. Pre-service teachers could also use Scratch—a programming environment that allows students to create programs by dragging and dropping blocks representing core constructs—to create simple programs and animations.

Recognizing the need for teachers to address computational thinking in their curricula and practice, several organizations, including the CSTA, ISTE, and the National Science Teachers Association, are also developing and sharing tools and resources for current and future teachers. Google’s Exploring Computational Thinking website (<http://g.co/exploringCT>) provides more than 130 lesson plans and sample programs aligned with international education standards; a collection of videos demonstrating how computational thinking concepts are used in real-world problem solving; and a “Computational Thinking for Educators” online course (<http://g.co/computationalthinking>). Since 2014, the Computer Science Education Research Group at the University of Adelaide in Australia has been partnering with Google to create introductory courses for implementing Australia’s Digital Technologies Curriculum and teaching computer science and computational thinking at primary and secondary levels, explicitly tied to the Australian curriculum (<https://csdigitaltech.appspot.com>). These resources provide a starting point for teacher educators to incorporate computational thinking ideas and relate them to specific subject area pre-service teachers will go on to teach in their future classrooms.

Conclusion

The 21st century is heavily influenced by computing, making it imperative that teacher educators incorporate computational thinking into elementary and secondary education. This means they must prepare teachers for computational thinking,² empow-

ering them to teach students these higher-order-thinking skills. Teacher-education programs are the opportune time to engage teachers early in their preparation to formulate ways to integrate computational thinking into their practice. Educational-technology and methods courses in elementary and secondary teacher preparation programs are ideal places for teacher educators to discuss computational thinking. The accompanying table summarizes our recommendations for teacher educators to embed computational thinking into teacher-education programs.

In summary, we have emphasized the importance of embedding computational thinking curricula in teacher education and provided recommendations for how teacher educators might be able to do it. For this effort to succeed, however, computer science and education faculty must work collaboratively, as both groups bring complementary expertise in computing and teacher development. **C**

References

- Ball, D.L. Breaking with experience in learning to teach mathematics: The role of a pre-service methods course. *For the Learning of Mathematics* 10, 2 (June 1990), 10–16.
- Barr, D. et al. Computational thinking: A digital age. *Learning & Leading with Technology* (Mar./Apr. 2011), 20–23.
- Barr, V. and Stephenson, C. Bringing computational thinking to K–12: What is involved and what is the role of the computer science education community? *ACM Inroads* 2, 1 (Mar. 2011), 48–54.
- Bundy, A. Computational thinking is pervasive. *Journal of Scientific and Practical Computing* 1, 2 (2007), 67–69.
- Calao, L.A. et al. Developing mathematical thinking with Scratch: An experiment with 6th grade students. In *Proceedings of the Design for Teaching and Learning in a Networked World 10th European Conference on Technology Enhanced Learning* (Toledo, Spain, Sept. 15–18). Springer International Publishing, 2015, 17–27.
- College Board. *AP Computer Science Principles*, 2014; <https://advancesinap.collegeboard.org/stem/computer-science-principles>
- Common Core State Standards Initiative. *Common Core State Standards for English Language Arts & Literacy in History/Social Studies, Science, and Technical Subjects*, 2010; http://www.corestandards.org/wp-content/uploads/ELA_Standards.pdf
- Computer Science Teachers Association. *Bugs in the System: Computer Science Teacher Certification in the U.S.*, 2013; https://cymcdn.com/sites/www.csteachers.org/resource/resmgr/CSTA_BugsInTheSystem.pdf
- Cooper, S. et al. *The Importance of Computing Education Research*. White paper, Computing Community Consortium, Jan. 14, 2016, 1–12; <http://cra.org/ccc/wp-content/uploads/sites/2/2015/01/CSEdResearchWhitePaper2016.pdf>
- Cszmadia, A. et al. Computational thinking: A guide for teachers. Computing at School Community, 2015, 1–18; <https://community.computingatschool.org.uk/resources/2324>
- Czerwinski, B.C. and Lyman, E.W. Exploring issues about computational thinking in higher education. *TechTrends* 59, 2 (Mar. 2015), 57–65.
- Darling-Hammond, L. and Bransford, J., Eds. *Preparing Teachers for a Changing World: What Teachers Should*

- Learn and Be Able to Do*. Jossey-Bass, San Francisco, CA, 2005.
- Denning, P.J. The profession of IT beyond computational thinking. *Commun. ACM* 52, 6 (June 2009), 28–30.
- Franklin, D. Putting the computer science in computing education research. *Commun. ACM* 58, 2 (Feb. 2015), 34–36.
- Grover, S. and Pea, R.D. Computational thinking in K–12: A review of the state of the field. *Educational Researcher* 42, 1 (Jan. 2013), 38–43.
- Hemmendinger, D. A plea for modesty. *ACM Inroads* 1, 2 (June 2010), 4–7.
- Kahney, H. *Problem Solving: Current Issues*. Open University Press, Buckingham, U.K., 1993.
- Lee, I. et al. Computational thinking for youth in practice. *ACM Inroads* 2, 1 (Mar. 2011), 32–37.
- Mishra, P. and Koehler, M.J. Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record* 108, 6 (June 2006), 1017–1054.
- National Research Council. *A Framework for K–12 Science Education: Practices, Crosscutting Concepts, and Core Ideas*. The National Academies Press, Washington, D.C., 2012.
- National Research Council. *Report of a Workshop of Pedagogical Aspects of Computational Thinking*. The National Academies Press, Washington, D.C., 2011.
- National Research Council. *Report of a Workshop on The Scope and Nature of Computational Thinking*. The National Academies Press, Washington, D.C., 2010.
- Repenning, A. et al. Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education* 15, 2 (May 2015), 1–31.
- Shulman, L.S. Those who understand: Knowledge growth in teaching. *Educational Researcher* 15, 2 (Feb. 1986), 4–14.
- Taylor, R.P. Introduction. In *The Computer in School: Tutor, Tool, Tutee*, R.P. Taylor, Ed. Teachers College Press, New York, 1980, 1–10.
- The Royal Society. *Shut down or restart? The way forward for computing in UK schools*. The Royal Society, London, U.K., Jan. 2012; <https://royalsociety.org/-/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf>
- Weintrop, D. et al. Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology* 25, 1 (Feb. 2016), 127–147.
- Wing, J.M. Computational thinking. *Commun. ACM* 49, 3 (Mar. 2006), 33–35.
- Yadav, A. et al. Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education* 14, 1 (Mar. 2014), 1–16.
- Yadav, A. et al. Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (Dallas, TX, Mar. 9–12). ACM Press, New York, 2011, 465–470.
- Yadav, A. and Korb, J.T. Learning to teach computer science. *Commun. ACM* 55, 11 (Nov. 2012), 31.

Aman Yadav (ayadav@msu.edu) is an associate professor in the College of Education and director of the Masters of Arts in Educational Technology program at Michigan State University, East Lansing, MI.

Chris Stephenson (stephensonc@google.com) is the head of Computer Science Education Strategy at Google, Mountain View, CA.

Hai Hong (haihong@google.com) leads the K–12 Education U.S. Outreach team at Google, Mountain View, CA.

Copyright held by the authors.

Watch the authors discuss their work in this exclusive Communications video. <http://cacm.acm.org/videos/computational-thinking-for-teacher-education>