Computer Science and Literacy at Overton Public School: A Scope and Sequence

Chase T. Christensen

The College of St. Scholastica

Abstract

Students at Overton Public School in Overton, Nebraska receive a total of over five hundred contact hours of compulsory technology education throughout the first ten years of their education. Until now, there has not been a concerted effort to define “technology,” much less organize these hours into a comprehensive scope and sequence. By undertaking and implementing this plan, Overton’s Kindergarten through 9th grade students will be guided through a developmentally-appropriate learning program that builds competencies in the areas of basic computer literacy, touch-typing, productivity and creativity suites, computer science, computational and design thinking, and digital citizenship. The primary objective of this program is to prepare these students for a life after high school which will require them to be technologically-literate, and able to creatively solve problems using efficiency and collaboration. This work presents a customized scope and sequence that is aligned to the 2017 Computer Science Teachers Association Standards and the State of Nebraska K-12 Technology Scope & Sequence and sources for lesson and assessment materials. This document is presented in partial fulfillment of the requirements for the Graduate Certificate in Computer Science Education at The College of St. Scholastica in Duluth, Minnesota.

Computer Science and Literacy at Overton Public School: A Scope and Sequence

As the 21st century progresses, it is becoming increasingly necessary for primary and secondary schools to evaluate and adjust their learning programs in response to the rapidly changing world in which they exist and into which they will discharge new generations of leaders and creators. Given the ubiquity of technology in all industries and its cemented status as an industry unto itself – as well as the ever-mounting pressure on institutions of education at all levels to adequately prepare students to drive innovation, often in competition with each other – schools which have not formalized their technology learning programs operate at a severe deficit of efficacy.

In keeping with the rapid pace of growth and change within the technology field in particular, it follows that the supporting curriculum in the schools must be as close to the cutting edge as is feasible and responsible given the most recently validated educational research. To that end, this document presents the following:

- an evaluation of the current technology curriculum in the context of one school system;
- a review of relevant research to determine the rationale for any proposed changes, including applicable state and national standards;
- a non-exhaustive list of lesson materials and assessment tools and their suggested uses as aligned with the above body of research; and
- a customized scope and sequence that leverages knowledge of sound educational practices, student and community strengths and interests, faculty expertise, schoolwide improvement objectives, financial investments and technology resources, and allotted instructional time to achieve positive learning outcomes for all students.

Introduction

Local Context

The Village of Overton is centrally located in Nebraska along Interstate 80. In 2010, the population of Overton was 594 people, exhibiting a racial makeup of 84.7% White, 10.4% Hispanic or Latino, and 4.9% other races or ethnicities. The local economy is supported largely by farming and livestock operations of various sizes. However, a majority of Overton residents commute to neighboring Lexington or Kearney for various other employment opportunities, including in the healthcare, education, banking, government, and retail sectors. The community's proximity to these population centers provides an economically diverse (if racially homogenous) population from which the local school district draws its students and its funding. Option enrollment (i.e. students who live outside of the district but choose to attend school in Overton) makes up 16% of the total student population, indicating that Overton is a desirable place for families to seek education for their children.

Overton Public School is the hub of the village – a centrally-located, single-building school district that employs 30 teachers to serve the educational and developmental needs of over 300 students annually in grades Pre-K through 12. The mission of the school is "to provide opportunities for everyone to be engaged, empowered, and enlightened." To this end, the school has implemented programs to enable students to become well-rounded citizens who are prepared for life after their high school graduation. This life most often includes directly entering the labor force or enrolling in trade schools, university degree programs, or pre-professional programs.

Infrastructural Context

The primary tenet of the philosophy of technology held by district leadership – including the school board, superintendent, principal, and technology director (who is the author) – is that a

generalist approach to student technology competency is preferred to a specialist approach. That is, given the diversity of technology ecosystems likely to be encountered post-graduation, the school environment should mirror that diversity such that all students are able to complete a variety of computing tasks using a variety of computing devices. Similarly, teachers should be allowed flexibility and freedom of choice both in the ways they incorporate technology into their curriculum and in the computing devices they use to do so. In fact, school administrators are in such support of this goal that they have an unwillingness to be swayed by the potential cost savings of homogenizing the school's technology infrastructure. This has resulted in a "mixed environment" of iPads, Chromebooks, Windows desktops and laptops, iMacs, and MacBooks – all of which are used on multiple occasions by every student throughout their education at Overton Public School.

In 2012 the district adopted the 1:1 iPad Initiative, which committed an Apple iPad to every student to support and enhance their learning. Each student in Pre-K through grade 6 is assigned an iPad which remains in a cart over nights, weekends, and breaks; each student in grades 7 through 12 is assigned an iPad which is taken home each night. All teachers are expected to incorporate use of the iPad into their instruction where developmentally and educationally appropriate. While the diversity of technology is treasured, this decision was made to create a minimum standard for cross-curricular technology implementation, and allow for portability of textbooks and assignments.

In addition to iPads, two mobile laptop carts are available for teacher and student use, one of which contains a full classroom set of Apple MacBooks, and the other, a full classroom set of HP Chromebooks. Both of these carts are used regularly at all grade levels for standardized testing, writing and creative projects, typing practice, and computer science lessons. A second

classroom set of Chromebooks is shared between the high school English and History teachers, both of whom employ extensive writing assignments that require the use of physical keyboards.

Two separate and fully-equipped computer classrooms exist at Overton Public School. One of these functions as the 7th through 12th grade business and technology classroom; it contains Windows-based desktop PCs. The other contains Apple iMacs, and it functions as the 5th and 6th grade technology classroom, but remains open for other classes to use throughout the day. There are additional smaller computer labs in the special education wing and industrial technology shop (iMac) and the library (Windows PC).

Teachers at Overton Public School are provided with an iPad for instructional use and are allowed a choice between a Windows or Apple laptop computer to complete their work. Many teachers also keep a Windows or Apple desktop computer in their classroom. All classrooms are equipped with a SMART Board and digital projector, as well as an Apple TV to encourage the sharing of and collaboration over student iPad or MacBook work. Available for scheduled use are two 3-D printers, a 360-degree camera, a set of thirteen Dash and thirteen Dot robots and accessories, and a classroom set of Adafruit Circuit Playgrounds with some Arduino boards.

Bearing in mind that the school serves less than 350 students in total, it may seem as though technology integration at Overton Public School needs no further optimization. However, it would be a dangerous assumption to make that the mere presence of technology effectuates the deep and diverse understandings or competencies intended by stakeholders and described above.

Curricular Context – Technology Faculty

At present, Overton Public School employs three faculty members (all of whom are White males) who teach technology-related curriculum on a part-time basis. In all cases, the teachers' primary content area focus is not technology related.

Teacher A (who is the author) is trained in music education and serves as the school's instrumental music director and technology director. The latter responsibility includes the strategic planning, implementation, and upkeep of all technology at the school. To an extent, this also includes training and coaching faculty and staff on the use of instructional technology. Finally, this position is responsible for the development and teaching of "Technology" classes at the elementary level – grades K through 4, and oversees 8th grade homeroom.

Teacher B is trained in social sciences education and has a background in educational technology management similar to Teacher A's current responsibilities. This teacher serves as the middle school social studies teacher, high school weights and conditioning instructor, and 5th and 6th grade "Computers" teacher.[1]

Teacher C is trained in business, marketing, and information technology education. This individual is the only one of the three to possess an official technology-related field endorsement. He is the yearbook coordinator and teaches business courses, and is also responsible for teaching 7th and 8th grade "Computers" and 9th grade "Info Tech" courses.

Curricular Context – Course Sequence and Descriptions

All explicit technology education at Overton Public School takes place between Kindergarten and 9th grade (inclusive). At all grade levels, this program of study is compulsory; that is, every student who attends the school must enroll in and pass every technology course offered. While this alleviates recruitment challenges, it does not automatically ensure that the education received is engaging, inclusive, and equitable; nor does it insinuate that students will meet learning targets or even that the course sequence is aligned with itself or with national or state standards. Finally, all learning that can be considered "technology" or "computer" related is intended to take place within this sequence of courses. Specific definitions of technology-related

[1] Teacher B is new to the district (2020-2021). References to his two courses below reflect those of his predecessor, who taught in the district for several decades.

learning domains will be explored later in this document. At present, the number of contact hours in the entire sequence amounts to approximately 500.

Title: Elementary Technology – Kindergarten	**Instructor:** Teacher A
Total Contact Hours: 9 hours	**Class Schedule:** One 30min. lesson per week, spring semester only
Description: This course exposes students to basic concepts of digital citizenship and computer science through the use of iPads and the Dot robot from MakeWonder, as well as several unplugged activities. Digital citizenship concepts include media balance and online safety (Common Sense Grade K). Computer science concepts include drag-and-drop programming, basic sequences and loops, and algorithmic thinking (Code.org CSF Course A).	

Title: Elementary Technology – Grade 1	**Instructor:** Teacher A
Total Contact Hours: 9 hours	**Class Schedule:** One 30min. lesson per week, spring semester only
Description: This course continues student exposure to basic concepts of digital citizenship and computer science through the use of iPads and Dash & Dot robots, as well as several unplugged activities. Digital citizenship concepts include online safety and technology-related SEL (Common Sense Grade 1). Computer science concepts include more complex sequences, basic loops and events, and debugging (Code.org CSF Course B).	

Title: Elementary Technology – Grade 2	**Instructor:** Teacher A
Total Contact Hours: 18 hours	**Class Schedule:** Two 30min. lessons per week, spring semester only
Description: This course marks a new stage of digital citizenship and computer science learning through the use of iPads and Dash & Dot robots, as well as several unplugged activities. This course also introduces touch-typing with MacBooks or Chromebooks. Digital citizenship concepts include online communities, media balance, private information/digital footprint, cyberbullying, and copyright (Common Sense Grade 2). Computer science concepts include more complex loops and events, basic binary code, and basic design thinking applied to a video game and computer art (Code.org CSF Course C). Touch-typing lessons introduce posture, hand position, and the home row, with at least 6 and no more than 10 contact hours devoted to this practice (TypingClub).	

Title: Elementary Technology – Grade 3	**Instructor:** Teacher A
Total Contact Hours: 18 hours	**Class Schedule:** Two 30min. lessons per week, fall semester only
Description: This course continues digital citizenship and computer science learning through the use of iPads and Dash & Dot robots, as well as several unplugged activities that introduce a competitive element. This course also continues touch-typing with MacBooks or Chromebooks. Digital citizenship concepts include password security, identity and self-expression, technology behavior norms, cyberbullying, and photo doctoring (Common Sense Grade 3). Computer science concepts include nested loops, events, simple and compound conditionals, binary code, and problem decomposition, debugging, and basic abstraction (Code.org CSF Course D). Touch-typing lessons continue where the class left off last year, with at least 8 contact hours devoted to this practice (TypingClub). The classroom teacher often chooses to continue this practice throughout the spring semester once per week.	

Title: Elementary Technology – Grade 4	**Instructor:** Teacher A
Total Contact Hours: 18 hours	**Class Schedule:** Two 30min. lessons per week, fall semester only
Description: This course continues digital citizenship and computer science learning through the use of iPads and Dash & Dot robots, as well as several unplugged activities. This course also includes a creative coding project. Digital citizenship concepts include media habits, private information, digital footprint, respectful gaming, responsible research, and accessible technology (Common Sense Grade 4). Computer science concepts include simple and compound conditionals, objects with parameters, and procedures; the final project incorporates design thinking and the problem-solving process (Code.org CSF Course E). After the completion of this project, the class experiences basic text-based programming in the Swift Playgrounds app (Apple LTC 1).	

Title: Computers – Grade 5	**Instructor:** Teacher B
Total Contact Hours: 90 hours	**Class Schedule:** Alternating two/three 50min. class periods per week, all year
Description: This course takes place in the iMac computer lab/classroom. Learning in this course has focused on touch-typing and the Microsoft Office productivity software suite, but has included exposure to Sketchup 3-D modeling software as well as Hour of Code activities.	

Title: Computers – Grade 6	**Instructor:** Teacher B
Total Contact Hours: 90 hours	**Class Schedule:** Alternating two/three 50min. class periods per week, all year
Description: This course takes place in the iMac computer lab/classroom and builds on the activities completed in the 5th grade course, going further in-depth on touch-typing and the Microsoft Office productivity software suite. Advanced Sketchup techniques are learned along with exposure to the Alice and Blender 3-D modeling software.	

Title: Computers – Grade 7	**Instructor:** Teacher C
Total Contact Hours: 45 hours	**Class Schedule:** One 50min. class period per day, first or second quarter only
Description: This quarter-long course takes place in the PC lab, and continues to practice typing skills on Typing.com, focusing on accuracy over speed. Brief explorations of the Microsoft Office productivity suite are conducted with emphasis on business applications.	

Title: Computers – Grade 8	**Instructor:** Teacher C
Total Contact Hours: 45 hours	**Class Schedule:** One 50min. class period per day, third or fourth quarter only
Description: This quarter-long course takes place in the PC lab, building on the skills learned in the 7th grade course with attempts to build typing speed. Additional explorations of the Microsoft Office productivity suite are conducted with emphasis on business applications.	

Title: Homeroom – Grade 8	**Instructor:** Teacher A
Total Contact Hours: 90 hours	**Class Schedule:** Two or three 50min. class periods per week, all year
Description: This course traditionally serves as a study hall or library access time, and allows for general housekeeping and grade checks. One day per week, the class meets with the school counselor for an SEL lesson. In an attempt to wean 8th grade students off of this "free" period (in preparation for high school), the class is given at least one weekly study hall period, and the two or three remaining days of the week are spent going through the Code.org Computer Science Discoveries curriculum, which provides students with unplugged problem-solving activities, has them build websites and JavaScript games, and program circuit boards.	

Title: Information Technology Applications I & II	**Instructor:** Teacher C
Total Contact Hours: 90 hours each	**Class Schedule:** One 50min. class period per day, each semester
Description: Both of these courses are completed in the 9th grade (one first semester, the other second semester). This course is a graduation requirement. The main emphasis of the first semester is learning/reviewing the keyboard, with the goal of building accuracy. After keyboarding skills are learned, emphasis is placed on creating letters/envelopes, manuscripts, tables, and other business correspondence. In addition, proofreading, spelling, and language skills are implemented. Timed writings are given once per week; however, accuracy is emphasized over speed. An extensive career unit is covered in the second semester.	

Rationale

In total, ten school years (Kindergarten through 9th grade) and approximately 500 contact hours encapsulate the current compulsory technology curriculum at Overton Public School. At all levels, including preschool (both 3-year-old and 4-year-old classes) and 10th through 12th

grade, technology is used to achieve non-technology learning targets. Students are routinely expected to use what they have learned about technology to complete assignments and show their learning. This involves creating multimedia presentations, conducting online research, composing original essays, and communicating and collaborating with others through technology. Given the pervasiveness of technology in the core curriculum and its importance in life after high school, it is necessary to analyze the effectiveness of the technology learning program as it exists now, understand the constituent domains of a strong technology curriculum, and then settle on appropriately descriptive terminology. In doing so, a rationale for the inclusion of the various domains in a technology curriculum may be established, leading to a shared vision among school technology leadership and ensuring a pathway to enduring understandings.

Analysis

A program of study that spans an entire decade should be well-organized to do the following:

- increase in rigor as students progress through it, deepening the learning of previously-encountered concepts while probing new aspects of the content area at each level;
- pique interest at key points in development to effectuate engagement and self-esteem in the content area;
- culminate in a learned body of knowledge and skills that prepares well and encourages students to pursue a career in a field related to the content area if desired.

In a technology learning program in particular, the concept of "transfer" – that is, learning in one context and applying it to another – is especially paramount. According to P.R.J. Simons, "There are three kinds of transfer: from prior knowledge to learning, from learning to new learning, and from learning to application" (Simons, 1999). The most effective technology curriculum will

result in all students successfully achieving all three types of transfer – not only within the course progression itself, but to all other curricular areas as well. This must occur before the student has even completed the entire curriculum. Due to the fact that all teachers of non-technology content areas are expected to incorporate student technology use into their curricula to some extent, these teachers would be served well if they knew the "what" and "when" of what their students were learning in their technology courses. All of this proves the need for a technology curriculum that is aligned with itself.

At present, the technology curriculum at Overton Public School appears to be aligned in three separate ways – each according to the teacher at that particular grade level. Kindergarten through 4th grade courses are well-aligned and sequenced, using five separate sources for lesson materials: Code.org, Common Sense Media, Apple Learn to Code, Dash & Dot robots, and TypingClub. The vast majority of the activities of these courses are facilitated by Code.org, Common Sense Media, and Dash & Dot robots – all three of which are explicitly divided into a K-4 sequence that is fairly easy to implement and adjust given the available resources and time constraints respectively.

The 5th and 6th grade classes reflect a stark difference in teaching style and philosophy, in addition to the classroom environment in general. Given that the instructor of this course retired after several decades in the district teaching social sciences and these two technology courses, the 2020-2021 school year represents both a loss of significant experience and a prime opportunity for necessary change. The previous instructor focused his courses heavily on typing skills and the Microsoft Office productivity suite, but he also enjoyed incorporating digital media production into these courses. This would take the form of publishing tools such as Comic Life, video production tools such as Final Cut Pro, or 3D modeling tools such as Alice and Blender.

Unfortunately, this instructor was resistant to changes in hardware and software (and curriculum, to an extent). This meant that until as late as his retirement at the end of the 2019-2020 school year, students in these classes were still using iMacs from 2010, which were running OS X Mavericks with the 2008 edition of Microsoft Office (and older versions of the other apps mentioned above). The obsolescence of this hardware and software resulted in frequent failures, slow performance, and a student population that could not say they were well-served by their school. Even most web-based tools were unavailable to these classes, because of the old browsers' lack of support for HTML5. Perhaps most importantly, through March 2020, students in the 5th and 6th grade computer classes were disengaged from the subject and routinely stated it was one of their least favorite classes. All computers in this classroom have been updated to new models for the 2020-2021 school year with Microsoft Office 2019, and a new teacher is ready to change this perception.

The 7th and 8th grade computer classes take place in the PC lab, which has also at times suffered from sluggish hardware and obsolete software. These classes have been using Microsoft Office 2010 since its release, but use modern browsers and operating systems. Due to a repeated emphasis on the same skills from 5th and 6th grade, students in this class can be forgiven for their similar lack of engagement in the subject matter. For the 2020-2021 school year, all computers in this classroom have been updated to new models with Microsoft Office 2019.

The 8th grade homeroom course represents the most recent effort to reinvigorate technology education at Overton Public School, especially with respect to the computer science gap that initiates after 4th grade. In the two years since the Computer Science Discoveries course has been implemented, a majority of survey respondents have agreed that they have the ability to learn computer science, they feel comfortable in and like this computer science class, and they

think computer science is interesting. This is convincing evidence that if computer science was implemented more broadly in the technology curriculum, it could lead to greater engagement in all technology-related classes.

Finally, in the 9th grade technology course, the emphasis on business applications for computer skills has the potential to pique the interest of a greater number of students, but it is still the case that the tasks assigned to students do not differ significantly from what they have experienced in middle school. By this point in a student's technology education, many students decide (or have already decided) that they do not enjoy working with computers, and could not see themselves studying computers in-depth beyond high school or working in a computing-related field.

This analysis of the current technology curriculum at Overton Public School reveals the following:

- At the K-4 level, learning targets revolve around respectful, responsible, and safe use of computing devices and the Internet, as well as basic programming skills and algorithmic thinking.
- At the 5-8 level of traditional technology courses, learning targets revolve around the clerical applications of computer skills, namely typing and word processing.
- At the experimental 8th grade level, students return to the building of computational thinking skills and problem-solving processes that characterized learning at the K-4 level, with emphases on job prospects and future opportunities in computer science.
- At the 9th grade level, learning targets return to clerical applications, and the course evolves into an introduction to business.

- Engagement in the curriculum and students' perceptions of the technology field suffer greatly under the current system. This ensures that though students may graduate with the ability to type and use obsolete productivity software, they do not find technology to be particularly fulfilling or useful, and do not consider a career in the field. Since May 2017, only one graduate (a white male) has gone on to study in an information technology field.

Technology Learning Domains

The field of technology is broad and diverse, such that it is common for stakeholders to exhibit misconceptions about what should be learned about it in schools and what actually is being learned. The *K-12 Computer Science Framework* quotes a 2007 report of the Computer Science Teachers Association (CSTA) Certification Committee:

> Many states did not seem to have a clear definition or understanding of the field 'Computer Science' and exhibited a tendency to confuse Computer Science with other subject areas such as: Technology Education/Educational Technology […] Industrial or Instructional Technology […] Management Information Systems […] or even the use of computers to support learning in other subject areas. (K-12 Computer Science Framework, 2016, p. 13)

This explains, in part, the "Wild West" that has been technology education at Overton Public School and other K-12 institutions across the United States. If policymakers, administrators, and teachers do not understand the difference between computer science and other forms of learning and doing with technology, there is little hope that students will exit high school sufficiently prepared or encouraged to choose a technology-related career path.

The Oxford English Dictionary defines the word "domain" as "a specified sphere of activity or knowledge" (Domain, 2020). Given the breadth and diversity of technology, but that

it is often ill-defined in a curricular setting, it is appropriate to use this word to describe the various spheres of activity and knowledge that comprise a complete education in the technology field. In order to improve the current state of technology education at Overton Public School, it is necessary to differentiate between these domains of technology learning – and in so doing, establish their purposes and justifications, define success, and prepare the curricular landscape for a more organized approach. According to the *K-12 Computer Science Framework*, "Computing education in K-12 schools includes computer literacy, educational technology, digital citizenship, information technology, and computer science" (K-12 Computer Science Framework, 2016, p. 13). The document goes on to define each term:

> As the foundation for all computing, computer science is "the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society" […]
>
> - Computer literacy refers to the general use of computers and programs, such as productivity software. […]
> - Educational technology applies computer literacy to school subjects. […]
> - Digital citizenship refers to the appropriate and responsible use of technology, such as choosing an appropriate password and keeping it secure.
> - Information technology often overlaps with computer science but is mainly focused on industrial applications of computer science, such as installing software rather than creating it. (pp. 13-14)

For the purposes of this scope and sequence, digital citizenship will stand on its own as a domain, educational technology will be incorporated into computer literacy, and information technology will be incorporated into computer science where appropriate.

Digital Citizenship. In his book *Digital Citizenship in Schools: Nine Elements All Students Should Know*, Mark Ribble differentiates between the concepts of "digital natives" (young people who have grown up around and seem to instinctively know technology) and "digital immigrants" (individuals who may have adopted new technologies as adults, but may not have an instinctive understanding). As users of technology have interacted with one another, Ribble argues that a new "digital" society has formed – one with its own set of behavioral norms and responsibilities. Though many digital immigrants might assume that digital natives already know everything there is to know about technology, those who identify with the former must recognize that "Even when students are comfortable using technology, they may not be using it appropriately" (Ribble, 2015, p. 1). Educators are reminded that nearly all of their students – at increasingly younger ages – are provided with smartphones, tablets, and web devices, but "little or no direction or professional development, or the training that is provided is incomplete, or incorrect. Too often the hope is that users will 'figure them out,' and this is when frustration and disenchantment occur, causing mistakes to be made" (p. 1).

Ribble goes on to outline nine elements of "digital citizenship," a field of study – or, rather, a state of being – that applies character education to digital technologies, and the resultant concept "reinforces the positive aspects of technology so that everyone can work and play in this digital world" (p. 7). These elements represent nine competencies that digital citizens possess: Access, Commerce, Communication, Literacy, Etiquette, Law, Rights and Responsibilities, Health and Wellness, and Security (pp. 16-17). Just as K-12 schools are entrusted with the duty of developing good citizens of a particular community, they should also take seriously their responsibility to train good digital citizens, especially as schools increasingly become the primary source of a child's exposure to technology at increasingly younger ages.

A successful implementation of a digital citizenship learning program is described in Ribble's book as one that "is not an add-on that can be addressed once or twice a year through assemblies. It is not a one-time vaccination, rather an integral part of [a school's] daily culture. It must be owned by all" (p. 75). This case study presented additional objectives and outcomes:

- Develop a clearly-articulated vision for technology as a tool for teaching and learning.
  - ⇒ Research and evaluate what technology in schools should look like.
  - ⇒ Critically analyze existing IT policy and consider re-writing it.
  - ⇒ Address how information about digital citizenship would be disseminated.
  - ⇒ Heavily involve students in the process of working on the district-level rights and responsibilities.
- Educate all stakeholders on the importance of being proactive, not reactive.
  - ⇒ Educate teachers on what technology as an educational tool looks like and how to develop teaching and learning opportunities for all students.
  - ⇒ Use community outreach to sensitize people as to why it is important to teach using technology, not ban it.
  - ⇒ Ask community partners what they would like students to have learned before entering the workforce.
- Increase student agency by including them in the creation of acceptable use guidelines.
- Invest in the technological infrastructure of the school.
- Continue to professionally develop teachers and support the use of technology as a learning tool.

The school's belief should be that "being a good digital citizen is one piece of the developmental puzzle that in the end will result in caring, collaborative, and creative adults" (pp. 75-79).

Computer Literacy. If the study of digital citizenship is concerned with how to respectfully, responsibly, and safely use technology, computer literacy (or technological literacy) is the study of how to use the technology in the first place. The International Technology and Engineering Educators Association defines "technological literacy" as "the ability to use, manage, assess, and understand technology" (Buckler, Koperski, & Loveland, 2018, p. 18). And according to Garmire and Pearson (2006) as quoted in Buckler, Koperski, & Loveland, "technologically literate people should have 'a basic knowledge about technology,' '[an ability to] employ an approach to solving problems that rely on aspects of a design process,' and the ability 'to think critically about technological issues and act accordingly'" (p. 18). These ideas represent an evolution in traditional technology education that has passed many educators by. The 21st century skills of critical thinking, problem solving, and design thinking must be woven into traditional computer literacy education just as they are into all other core subject areas.

As Ribble noted, the generation of digital natives appears to intuitively know how to use technology. This is likely due to the ubiquity of computing devices in their lives at home and at school. This begs the question of whether students actually need explicit computer literacy instruction – which hails from an era in which computers had their own rooms and workstations, and computing tasks could not practically be learned in any other setting. Indeed, many educational institutions have removed dedicated computer labs altogether, repurposing them into communal workspaces, flexible classrooms, robotics labs, or "makerspaces" fit for the type of hands-on or collaborative learning that is increasingly the norm. Ironically, these are likely to involve less technology, and could even explore the same human-centered design approaches that gave rise to a generation of computing devices so intuitive that their functions could be immediately understood by the toddlers who would grow up to occupy these very classrooms.

There are things that all students should know and be able to do with computer applications. Included in these are traditional touch-typing – defined as the ability to type on a QWERTY keyboard relying on muscle memory and not the sense of sight to find the keys – and basic functions of word processing, spreadsheet, and presentation software. However, even productivity software such as these can be learned quickly by digital natives just as thoroughly in their other subject areas. Word processors are a modern requirement of English Language Arts classes, spreadsheets have a natural place in finance classes or statistics units, and presentation software can be utilized in a speech class or any unit requiring students to show their learning.

Compounding the problem of computer literacy courses wasting instructional time on knowledge and skills that students have largely already attained is that if this persists, all students – but particularly women – are increasingly turned off from enjoying or wanting to continue in courses or careers related to information technology. According to a 2008 study, quoted in Kindsiko, Aidla, Poltimäe, and Türk, female students were much less likely to consider an information and communications technology (ICT) academic or career path if they perceived ICT subjects to be boring (Kindsiko, Aidla, Poltimäe, & Türk, 2020, p. 54). Kindsiko et al. draw attention to a "long stream of studies" that show engineering, science, and ICT fields are perceived as stereotypically male occupations, especially by young women (p. 54). While strong preparation at the high school level is necessary in these fields, this is not sufficient to effectuate student interest. Especially in the case of women, this falls to "the role of socializers in determining future academic and career choices – namely, those high school seniors who have decided to choose science and engineering careers seem to have experienced specific encouragement from parents or teachers" (p. 54). The authors thus assert that "in order to increase female interest in ICT related subjects, schools should diversify the way ICT courses are

taught because some ways might reinforce gender stereotypes and alienate young women by constructing computing as a masculine practice" (p. 55). This has as much to do with the classroom environment and the teacher's methods as it does the content being presented.

Kindsiko et al. point out that the ICT field is perceived to rely on "out-of-the-box" thinking and problem solving, but during the school years, "girls tend to adopt the 'familiar algorithmic reasoning' – sticking to the standard methods[…] Simply put, girls tend to follow what their teachers had shown them" (p. 56). Instructors should consequently break this pattern, and begin to use their computer literacy courses to break the pattern of learning mostly by the textbook. As alluded to above, the content of most traditional computer literacy courses is actually below the level most appropriate for the learners, and it is not connected to the way students use technology outside the school. It is known that linking the content of a course with real-life problems has a positive impact on student interest in a subject area; additional studies have also confirmed this to be the case in computer classes (p. 56). Some final conclusions of this study include:

- Regardless of a school's ICT infrastructure, an upgrade in teaching will not take place without highly-skilled teachers who see themselves as capable of using technology (p. 58).
- The more survey respondents agreed with the statement, "computer classes were interesting," the greater the odds of considering studies in an ICT field (p. 61).
- Students generally assess computer classes as being "rather outdated and too basic. […] this finding is in line with the rather poor ICT skills of computer class teachers" (p. 62).
- Students perceive that schools have not paid attention to the quality and rigor of ICT classes, which "kills their interest" (p. 62).

- Students value challenging tasks, and are not interested in content at a basic level (p. 63).
- In order to increase female interest in ICT subjects, "there should be much more attention paid to how computer classes are taught, and avoiding fostering gender stereotypes in class" (p. 65).

The culmination of this research points to a need for learning targets and activities that are diverse, personal to students' lives and interests, and commensurate with their current ability levels (which are almost always higher than estimated by schools). Computer literacy courses should be facilitated by instructors who perceive themselves to be capable of teaching the subject, and who can foster a classroom of out-of-the-box thinking that does not perpetuate gender stereotypes and prioritizes student engagement.

Computer Science. On December 17, 2018, Nebraska Commissioner of Education Matthew L. Blomstedt delivered a letter to all superintendents of public schools in Nebraska highlighting the growing need for computer science in the workforce and the requisite training at the K-12 level. He wrote that including computer science throughout K-12 settings "promises long term equity benefits," for those historically underrepresented in the field (Blomstedt, 2018). Blomstedt raises the important point that Nebraska's leading industries (healthcare, manufacturing, agriculture, banking, etc.) all require professionals with computer science skills, and he stresses that this instruction should not be isolated. He encourages "integrated approaches [that] simulate the authenticity of the workplace and better prepare our students for postsecondary education and high skill/high wage/high demand careers" (Blomstedt, 2018). Finally, Blomstedt affirms a new regulation that permits school districts to consider computer science as part of the core curriculum, and allow computer science courses to count toward graduation requirements.

As noted in the *K-12 Computer Science Framework*, computer science is itself a field of study separate from ICT or computer literacy as defined above. Fluck, et al. compile three popular definitions of computer science, presented in chronological order of publication:

- It seeks to answer the following questions: What is information? What is computation? How does computation expand what we know? How does computation limit what we can know? (Denning, 2007).
- The study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications,[2] and their impact on society. (Seehorn et al., 2011).[3]
- The scientific and practical approach to computation and its applications.[4] It is the systematic study of the feasibility, structure, expression, and mechanization of the methodical procedures (or algorithms) that underlie the acquisition, representation, processing, storage, communication of, and access to information. (Wikipedia, 2015). (Fluck, et al., 2016, p. 39)

For the basis of their research, Fluck et al. prefer the latter definition for its recency and detail. However, the second definition is believed to be more appropriate for the K-12 setting for its relative simplicity. This definition is also mirrored in the *K-12 Computer Science Framework*, which provides the national foundation for approaches to the development of computer science curriculum.

Fluck et al. present economic, social, and cultural rationales as their three-pronged argument for including computer science in the K-12 curriculum. They recognize that a nation must produce enough computer scientists to sustain "a competitive edge in a world driven by technology," but they precognize Blomstedt in their assertion that there is also a requirement "for

---

[2] Here, the term "applications" refers to the use to which computers and computation are put, rather than software.
[3] Seehorn et al. is the 2011 revision of the *CSTA K-12 Computer Science Standards*.
[4] See [(2)].

computer science-enabled professionals in all industries to support innovation and development" (p. 41).

At present, the United States system of education is unable to meet this need. According to a 2013 report by the Immigration Policy Center of the American Immigration Council, "the U.S. economy is capable of absorbing more high-tech professionals than the U.S. educational system produces. That is one reason so many U.S. scientists and engineers are immigrants" (Immigration Policy Center, 2013, p. 1). An earlier joint report by the Information Technology Industry Council, the Partnership for a New American Economy, and the U.S. Chamber of Commerce found that every foreign-born student who graduates with an advanced STEM degree from a U.S. university will create an average of 2.62 jobs for American workers. Given the large number of unfilled positions in nearly all STEM occupations, the report also suggested that these foreign-born workers are complementing and not displacing their U.S. counterparts (p. 2).

The Immigration Policy Center report promotes two other key positions. First, the demand for highly-skilled STEM workers is not being met, and is coming from "industries like Professional and Business Services, Healthcare Services, Advanced Manufacturing, Mining, and Utilities and Transportation" (p. 2). Finally, "To create the best possible science and engineering workforce, the United States must reform both its educational and immigration systems" (p. 2). As in the days of the Manhattan Project, the Cold War, and the Space Race, global superpowers are again competing for the brightest technological minds, and they can be either attracted through welcoming immigration policy or "home-grown" through a dedicated education system.

Fluck et al. provide as their social rationale for the inclusion of computer science in the curriculum the notion that active creators and producers of technology are more valuable to society than passive consumers of technology. A school which follows the traditional, ineffective

ICT learning model as outlined above will produce only passive consumers of technology, but a fully integrated computer science curriculum adds creativity, productivity, and innovation to the technology skills of all students. Perhaps most important for a society grappling with the rapid adoption of new technologies, Fluck et al. believe, "Rather than being oppressed by innovation shock, a society equipped with its own creative proponents of new ideas is more likely to sift them and control their impact" (Fluck, et al., 2016, p. 41). This, they claim, is a strong argument for teaching computer science in schools.

Finally, Fluck et al. paint computer science as a field which may be used to transmit various aspects of a nation's culture to other parts of the world – or to prevent cultural changes from being imposed upon a nation through technological developments from outside their society. The exporting of movies from Hollywood to the rest of the world is given as an example of the language, customs, attitudes, ethical values, and mores "which reflect the USA context" that may be transmitted to other societies (p. 41). Given that these cultural attributes are not universally shared, other styles of filmmaking originated in Asia and Africa as a way to preserve that culture which already existed. Video games and social media – both of which require extensive computer science expertise – are seen as the new frontiers in transmitting culture across the globe, or preserving it at home, and this represents the third rationale for including computer science in the curriculum (p. 42). Along with these new frontiers come also new cultural aspects of a digital society, such as data privacy and the Internet as a public utility.

If it is therefore settled that computer science must be included in the curriculum, it remains to be decided how this should be accomplished. Blomstedt encouraged Nebraska educators not to "isolate" computer science instruction, and it is known that school districts are stretched increasingly thin with the financial, temporal, and human resources they have. It would

seem that as much computer science learning as possible should occur without drastically altering the daily schedule or catalog of course offerings. Fluck et al. refer to this approach as "integration" – analogous to the interweaving of general literacy and numeracy into many subject areas of the school curriculum. This approach is seen as desirable, particularly at the elementary level. True integration would involve existing elementary teachers incorporating computer science skills – or at least computational thinking skills – within the other subjects they teach.

As noted, however, schools and teachers are stretched increasingly thin with their expectations for standardized testing results and the curriculum they already have. A large-scale study of 37 countries concluded that "integration of ICT use into other subjects was spasmodic and ineffective" and, "information literacy in the Netherlands disappeared as a separate subject, because of poorly trained teachers and a vague place in the curriculum" (Fluck, et al., 2016, p. 43). These examples, according to Fluck et al., "indicate computer science could have a similar fate if only taught through integration" (p. 43). A middle ground clearly must be negotiated for computer science in the curriculum – perhaps one in which generalized computational thinking skills are integrated in all subject areas, but more explicit computer science experience is set apart and delivered by a highly-qualified computer science teacher.

Rationale in Context

When it comes to the potential for successfully providing a comprehensive technology education for all students, Overton Public School is resource-rich. Each teacher and student has physical access to diverse forms of technology that are now regularly maintained and updated. Funding sources for the purchase of new technology and to support technology curricula are secure and committed. Faculty members accept their responsibilities to incorporate technology

into all curricular areas and to do so in ways that encourage safe, respectful, and responsible behavior by all students. From grades Kindergarten through 9, up to 500 contact hours are dedicated specifically to the technology education of all students. A comprehensive technology curriculum that is aligned to state and national standards – and is responsibly organized to build a diversity of technology-related knowledge and skills – is the only missing component. Given this context, the following guiding resolutions are set forth:

- The technology education program at Overton Public School shall be comprised of lessons, activities, assessments, and experiences in the technology learning domains of digital citizenship, computer literacy, and computer science, as defined herein.
- This program shall be delivered to all students in grades Kindergarten through 9, and shall be compulsory. At all grade levels, students shall receive some combination of instruction in all three learning domains.
- Learning targets shall be derived from a combination of standards documents from both state and national educational organizations as described below.
- The freedom of teachers to design learning programs that enrich or extend these standards, or to select or create their own learning materials to support the same in accordance with good teaching practice, shall not be infringed.
- At all times, in all technology classes, teachers shall be mindful of their own implicit biases that have a tendency to create a less-equitable learning environment for female students and students of color. All learning materials shall be evaluated by the teacher for a propensity to induce stereotype threat or exhibit bias.
- Teachers shall make a good-faith effort to seek out and eliminate reasons that any student might feel unprepared or unwelcome to contribute to the digital society in which they live.

Learning Targets, Materials, and Teaching Methods

The standards movement in education is often debated in terms that pit centralized, national control of learning in schools against decisions made in a local context by education professionals who know the needs and values of the community. The marriage in educational policy of the standards movement with high-frequency, high-stakes testing has further complicated and politicized the debate. Setting aside the discussion of standardized testing, however, it is becoming clear that a standards-based curriculum is a positive development for schools who wish to prepare students for competency in a global economy.

The need for standards was articulated by Matthew Gandal in a 1995 issue of *Educational Leadership*:

> We want all kids to have access to a rich and challenging curriculum […] This isn't what's going on in schools today. Some children get exposed to rigorous courses; others don't. Some students only get good grades if they master challenging material; others get good grades […] no matter what they do. […] We think common, rigorous standards can help us turn that around. (Gandal, 1995)

Gandal believes that this system is unfair, particularly to those students who "coast" through the system only to find out later just how little they actually learned. Additionally, as the United States continues to have one of the highest rates of student mobility of all developed nations (Heinlein & Shinn, 2000), Gandal argues that, "With clear, common standards in place, teachers ought to know what their incoming students have learned, regardless of where they are coming from" (Gandal, 1995). This again highlights the interconnectedness of the modern world – a state of being that has increased exponentially since the article's initial publication.

Detailed standards – or sets of statements about what all students should know and be able to do in a given subject area – are a way to express universally-applicable targets of learning activities without infringing on the rights of states, districts, schools, and teachers to decide how the curriculum should be delivered and assessed within their own context. However, Gandal writes that if standards begin to focus merely on the abstract skills of "critical thinking" and "problem solving" – or similar concepts – in the absence of any subject matter, "It's impossible to figure out what students are supposed to learn or teachers should teach" (Gandal, 1995). Therefore, while teaching methods should promote these high-level cognitive activities, content-area standards themselves must be specific, measurable, and achievable.

Fortunately, the work of developing standards for the technology learning domains of digital citizenship, computer literacy, and computer science has been underway for several years; its fruits are sufficient to lay a foundation for a new schoolwide technology curriculum. The standards upon which this new curriculum will be based are sourced primarily from the *Nebraska K-12 Technology Scope & Sequence* (NDE, 2018) with support from the *CSTA K-12 Computer Science Standards* (CSTA, 2017) and other documents as cited. Also discussed are suggested learning materials and teaching methods to ensure successful implementation of the curriculum.

Digital Citizenship

Learning Targets. The *Nebraska K-12 Technology Scope & Sequence* (NETSS) enumerates one standard of "Copyright" competency ("Explain fair use guidelines for copyrighted material") and ten standards of "Responsible Use" competency, which include:

- explaining and complying with Acceptable Use Policies and classroom rules related to technology use and networks;

- identifying, explaining, and demonstrating strategies for the safe and efficient use of computers, email, and networked devices;
- explain the need for and identify different types of accessible technology;
- identifying cyberbullying and describing strategies to resolve such situations; and
- explore, recognize, and analyze social and ethical impacts of technology as well as the ways in which media and data can be used to distort information.

The International Society for Technology in Education (ISTE) more succinctly includes digital citizenship concepts as a strand in its Standards for Students:

> (2a) Students cultivate and manage their digital identity and reputation and are aware of the permanence of their actions in the digital world.
>
> (2b) Students engage in positive, safe, legal, and ethical behavior when using technology, including social interactions online or when using networked devices.
>
> (2c) Students demonstrate an understanding of and respect for the rights and obligations of using and sharing intellectual property.
>
> (2d) Students manage their personal data to maintain digital privacy and security and are aware of data-collection technology used to track their navigation online. (ISTE, 2016)

Finally, Ribble's nine elements of digital citizenship can also be used as a basis for the implementation of digital citizenship learning targets. All of these sources emphasize student awareness and consideration of the risks and benefits of engaging themselves in a digital society.

Learning Materials. The following learning materials are suggested resources to aid teachers in the delivery of content related to the digital citizenship standards:

Resource/Provider	Grades	Description
Common Sense Education	K-12	This comprehensive curriculum is aligned to Common Core ELA, CASEL, AASL, and ISTE standards for digital citizenship, media literacy, and social-emotional learning. Learning is divided into six areas: 1. Media balance and well-being 2. Privacy and security 3. Digital footprint and identity 4. Relationships and communication 5. Cyberbullying, digital drama, and hate speech 6. News and media literacy Each grade level is assigned 3-7 lessons, which are complete with videos, presentations, and activities.
Be Internet Awesome	3-8	This curriculum, developed by Google, is aligned to ISTE and AASL standards, and divides learning into five areas: 1. Be Internet Smart: Share with Care 2. Be Internet Alert: Don't Fall for Fake 3. Be Internet Strong: Secure Your Secrets 4. Be Internet Kind: It's Cool to Be Kind 5. Be Internet Brave: When in Doubt, Talk it Out Each lesson includes presentations and activities. The highlight of this curriculum is the interactive web-based game Interland, which can be used as an assessment tool.
Our Space	7-12	This casebook is a set of resources that allow teachers to facilitate conversations with young people about digital ethics, and is not a curriculum as much as a "toolkit" of activities. It is divided into five units: 1. Participation 2. Identity 3. Privacy 4. Credibility 5. Authorship and Ownership
bCyberwise Monster Family	5-8	This play-through iPad game requires players to complete tasks that help a monster family new to the neighborhood contend with a cyberbully and learn good digital citizenship habits.
Digiduck	K-2	This interactive e-picture-book includes reflection questions, sound effects, and optional narration; it tells the story of a young duck who learns to make good choices about what to post online.

Teaching Methods. Effective digital citizenship instruction requires knowledge of the risks and benefits associated with living in a digital society, as well as the ability to facilitate open and honest discussion with students about their technology use without the use of patronizing or scaremongering language. According to CyberWise, digital citizenship curricula must be proactive, in order to empower young people to use technology confidently and wisely; not fear-based, with preferences toward resources that "help young people learn how to harness the power of digital technologies in positive ways that prevent [cyberbullying, sexting, etc.] in the first place;" and behavior-focused, insisting that digital citizenship is about basic behaviors such as being nice, more than it is about using technology (CyberWise, 2020).

Computer Literacy

Learning Targets. The NETSS devotes the majority of its standards to computer literacy-related areas of knowledge and skills. Therefore, it is unnecessary to seek other sources for learning targets in this domain. These five areas are platform-agnostic and include:

- Basic Technology: including Keyboarding (WPM benchmark = 5 x grade level beginning at grade 2), File Management, Basic Device Functionality, and Hardware and Software
- Productivity Applications and Tools: including Word Processing, Spreadsheets, and Presentation Tools
- Digital Media
- Research
- Communications and Collaboration

As Kindsiko et al. revealed, most students have already achieved mastery of computer literacy learning targets before entering the ICT classroom. The relative simplicity of some of the NETSS standards confirms the likelihood of this assertion:

- "Turn on the computer" (NDE, 2018, p. 6).
- "Use a pointing device to click menus and icons" (p. 6).
- "Watch videos and use play, pause, rewind, and forward buttons" (p. 16).

If examples such as these formed the majority of computer literacy learning targets, the fully integrated approach explored in Fluck et al. could easily suffice, because all teachers and at least some students at all grade levels could be assumed to have already mastered them. However, it should be reiterated that the research reveals non-rigorous learning targets to be a contributing factor in the disengagement of students from the ICT field. More complex learning targets do exist in the NETSS, however, and necessitate either a computer literacy course of instruction or a dedicated unit in a non-ICT subject area wherein the primary learning target is ICT-related:

- "Troubleshoot basic hardware and software problems" (p. 7).
- "Describe the components and functions of computers and networks" (p. 7).
- "Apply advanced formatting and page layout features when appropriate" (p. 11).
- "Use Internet browsers and search engines and online directories, compare the differences, and explain how they disseminate information" (p. 22).
- "Identify and explain current hardware and software trends" (p. 23).
- "Plan and implement a collaborative project with other students using technology tools (email, discussion forums, video conference)" (p. 26).

When organizing these learning targets into a context-specific scope and sequence, it will be necessary to decide and clearly identify which targets should be expected to be integrated in non-ICT-related educational technology use and which should be addressed within an ICT course. These decisions will be informed by the context, but the stated goal is that these learning targets

will be integrated as much as possible. In so doing, more instructional time in dedicated ICT courses can be spent with computer science instruction and the practice of 21st century skills.

Learning Materials. The following learning materials are suggested resources to aid teachers in the delivery of content related to the computer literacy standards:

Resource/Provider	Grades	Description
TypingClub	2-12	This website provides a variety of videos and activities that reinforce key concepts of touch-typing, such as posture and ergonomics, hand placement, correct finger use, and speed and accuracy building. The site provides enough differentiated content to supply an entire multi-grade touch-typing curriculum.
KeyboardingOnline	1-12	This website features self-contained and scaffolded touch-typing courses, including: • A 50-60-hour Keyboard Mastery course • A condensed 20-30-hour course of the same material • A 20-30-hour course building numpad skills • A 20-30-hour keyboarding course for the elementary level • A 50-60-hour course introducing Microsoft Office applications
Applied Digital Skills	6-12	This curriculum from Google uses G Suite tools such as Sheets, Slides, Docs, Gmail, Calendar, and Drive to improve skills such as: • Data Analysis ⇒ Create spreadsheet formulas ⇒ Identify patterns in data ⇒ Visualize data using graphs • Research & Communication ⇒ Evaluate bias ⇒ Research a topic ⇒ Create presentations • Coding & Digital Literacy ⇒ Use digital tools ⇒ Implement algorithms ⇒ Debug code

eReadiness	5-12	This set of computer literacy curricula includes a plethora of learning paths and activities that provide students with exposure to and practice with various desktop applications from the perspective of a business user. Some curricula are platform-agnostic.
Computer Skills Curriculum for Adult Learners	9-12+	This Open Educational Resources curriculum is intended for adult learners, but does provide basic introductions to computers, websites, Internet safety, email, social media, and Microsoft Word and Excel.
Microsoft Digital Literacy	5-12	This self-paced, online curriculum covers basic computing information that aligns with some of the more information technology-oriented standards of the NETSS, such as hardware and software, the Internet, and the cloud.
TestOut	7-12	This website provides self-paced curricula used not only in schools, but in career training programs as well. Its strengths lie in a robust, fully-online simulation environment that assess specific computer literacy skills, from the use of productivity software to the hardware components of a computer. Completion of these courses prepares students for most industry-standard competency exams.
Apple Everyone Can Create	5-12	This set of courses from Apple is designed to familiarize students with the features of the iPad that allow them to express themselves and share their creations. The four courses are Drawing, Video, Photo, and Music.

Teaching Methods. In the survey of students conducted by Kindsiko et al., the statement that was most strongly disagreed with was, “The computer classes were taught in an inspiring manner;” only 28% of students agreed (Kindsiko, Aidla, Poltimäe, & Türk, 2020, p. 63). Regardless of the difficulty level of the ICT course, most students disagreed with all of the following statements as well: “Computer classes are popular among students,” “The level of teaching was high,” and “The classes were interesting” (p. 63). This study well establishes that the classroom experience of students in ICT courses is generally actively detrimental to students’ interest and growth in the field – especially for female students. Kindsiko et al. blame poor ICT teaching methods – specifically ones that perpetuate gender stereotypes. The ICT classroom can

benefit from research-based engagement practices that support diversity, and a collection of these was published online by the National Center for Women & Information Technology. This framework is comprised of three overarching principles of student engagement, each with its own set of specific practices:

- Make It Matter
  - ⇒ Use meaningful and relevant content
  - ⇒ Make interdisciplinary connections
  - ⇒ Address misconceptions about the field
  - ⇒ Incorporate student choice
- Build Student Confidence & Professional Identity
  - ⇒ Give effective encouragement
  - ⇒ Offer student-centered assessment
  - ⇒ Mitigate stereotype threat
  - ⇒ Provide opportunities for interaction with faculty
- Grown an Inclusive Community
  - ⇒ Avoid stereotypes
  - ⇒ Use well-structured collaborative learning
  - ⇒ Encourage student interaction (NCWIT, 2016)

Teachers may also discover that a renegotiated scope and sequence will integrate in other content areas many of the skills that students previously found to be uninspiring aspects of a computer literacy curriculum. This substantial adjustment to the learning targets of the ICT course provides opportunities for novel teaching methods to be experimented with, such as flipped learning or project-based learning.

Computer Science

Learning Targets. Despite Blomstedt's advocacy for computer science in the K-12 curriculum, the NETSS outlines only two areas of the computer science domain: seven computational thinking standards and four programming standards. The latter standards spell out broad and vague competencies: "Write programs using visual programming languages," "Create and modify animations," "Write programs using text-based languages," and "Create web pages with a practical, personal, and/or societal purpose" (NDE, 2018, p. 30). Given the important role that computer science plays in modern society, and the inadequacy of the K-12 system of education in preparing a computer science workforce, students and stakeholders at Overton Public School deserve a more clearly articulated set of computer science learning targets. It is for this reason that the *K-12 Computer Science Standards* of the CSTA are consulted.

These standards delineate four levels of learning targets (1A, 1B, 2, 3A) that all students should meet before graduating high school, and one level (3B) for schools wishing to offer an advanced computer science course as an elective. The *K-12 Computer Science Framework*, which gave rise to the first draft of the CSTA standards, outlines five core concepts (what students should know about computer science) and seven core practices (what students should do using computer science). The framework further defines each concept and practice as well as numerous sub-concepts and sub-practices. Each of the standards correlates to one core concept and at least one core practice.

Core Concepts	Core Practices
Computing Systems • Devices • Hardware and Software • Troubleshooting	1. Fostering an Inclusive Computing Culture 1.1. Include the unique perspectives of others 1.2. Address the needs of diverse end users 1.3. Employ self- and peer-advocacy
Networks and the Internet • Network Communication and Organization • Cybersecurity	2. Collaborating Around Computing 2.1. Cultivate working relationships 2.2. Create team norms, expectations, and equitable workloads 2.3. Solicit and incorporate feedback 2.4. Evaluate and select technological tools
Data and Analysis • Collection • Storage • Visualization and Transformation • Inference and Models	3. Recognizing and Defining Computational Problems 3.1. Identify complex, interdisciplinary, real-world problems 3.2. Decompose complex real-world problems 3.3. Evaluate whether it is appropriate and feasible to solve a problem computationally
Algorithms and Programming • Algorithms • Variables • Control • Modularity • Program Development	4. Developing and Using Abstractions 4.1. Extract common features 4.2. Evaluate and incorporate existing technological functionalities 4.3. Create modules and develop points of interaction 4.4. Model phenomena and processes and simulate systems
Impacts of Computing • Culture • Social Interactions • Safety, Law, and Ethics	5. Creating Computational Artifacts 5.1. Plan the development of a computational artifact 5.2. Create a computational artifact 5.3. Modify an existing computational artifact
(K-12 Computer Science Framework, 2016, pp. 74-83, 89-92)	6. Testing and Refining Computational Artifacts 6.1. Systematically test computational artifacts 6.2. Identify and fix errors systematically 6.3. Evaluate and refine a computational artifact
	7. Communicating About Computing 7.1. Select, organize, and interpret large datasets 7.2. Describe, justify, and document computational processes and solutions 7.3. Articulate ideas responsibly

In a 2013 column entitled, “Making Computer Science Count,” then-chief operating officer of Code.org, Cameron Wilson, wrote, “When courses are taught in this area or are part of

the curriculum, they are too often focused on teaching students simply how to use technology […] instead of how to create technologies" (Wilson, 2013, p. 33). The CSTA standards use the concepts and practices to tease out exactly what types of cognition and action are required to prepare students to be creators of technology in a world dominated by computer science. By contrast, the NETSS programming standards still prescribe specific types of computational activities for students to consume. These learning targets may involve programming, but they do so in a way that teaches students "simply how to use" programming languages, "instead of how to create technologies" with them.

Computational Thinking. Wilson writes further that "ensuring access to engaging and rigorous K-12 computer science education" must be the starting point for imparting students with computational thinking skills (p. 33). At first glance, this statement appears to be backwards. Is the end goal of the computer science education movement not actually a rigorous and engaging computer science education that will inspire a new generation to consider a computer science career? Alas, Wilson – one of the founding members of Code.org – believes "computational thinking skills" to be the true promised land of computer science education. Computational thinking is an abstract practice akin to problem-solving, critical thinking, creativity, and leadership. There are certainly explicit pieces of knowledge and specific practices that might pertain to each of these notions that would be assumed to make one "better" at them. However (recalling Gandal's assertion about abstract skills), as a vague standard of achievement unto itself, computational thinking could not be called a learning target at all, given that it provides no guidance on what a student should learn or what a teacher should teach.

It is simply not realistic to expect that every student who progresses through even the best of technology curricula should go on to choose a career in information technology or computer

science. There are, however, certain skills – computational thinking chief among them – that all students, regardless of future plans, should be expected to leave with. Angeli et al. write:

> CSTA and ISTE […] developed an operational definition of computational thinking as a problem-solving process that includes, but is not limited to, the following elements: (a) Formulating problems in a way that enables us to use a computer and other tools to help solve them; (b) Logically organizing and analyzing data; (c) Representing data through abstractions, such as models and simulations; (d) Automating solutions through algorithmic thinking (i.e. a series of ordered steps); (e) Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources; and (f) Generalizing and transferring this problem-solving process to a wide variety of problems. (Angeli, et al., 2016, p. 49)

None of these tasks require the use of computers, but they all require thinking about problems the way that a computer would. Krauss and Prottsman explain computational thinking more simply as "Using special thinking patterns and processes to pose and solve problems or prepare programs for computation. Notably decomposition, pattern matching, abstraction, and automation" (Krauss & Prottsman, 2017, p. 4). Much ado is made about problem solving as a skill and disposition that all students need, but problem finding and problem posing are equally valuable skills in a world in which computers tend to do much of the actual solving work. Training one's mind to articulate a problem and the set of steps that will lead to its solution(s) in terms that could be communicated to a computer is a crucial, novel skill that will make one better equipped to enter any industry in any role than they would be otherwise.

The research suggests that computational thinking skills are so critical to the development of a truly technologically-literate citizen of the digital age that it would actually be better for

students at the elementary level to engage in activities that build these skills than for them to participate in any activities that involve explicit programming. Put another way, a curriculum that emphasizes programming but not computational thinking does not benefit students as much as one that emphasizes computational thinking and includes no programming at all. For the best outcomes, Angeli et al. suggest "a curriculum for K-6 with an explicit focus on computational thinking, before covering more theoretical and applied concepts of computer science in secondary education" (Angeli, et al., 2016, p. 49). The framework the authors propose is divided into three grade bands (K-2, 3-4, 5-6), wherein children are engaged in "thinking and problem solving by developing a solution to a problem, automating the solution through algorithmic thinking, and generalizing this solution to new problems when common patterns are identified or recognized" (p. 50). This computational thinking framework, as well as the learning materials and teaching methods broached by Krauss and Prottsman, should be consulted in the development of a technology education scope and sequence at the elementary level.

Learning Materials. Given the above conclusions that extensive exposure to and comfort with computational thinking skills and concepts should be considered a prerequisite to more explicit programming, the merits of "unplugged" computer science activities should be briefly discussed. Cortina writes about the "CS Unplugged" lesson materials which exemplify "experiential learning, where participants learn through activity outside of a standard academic setting. […] CS Unplugged activities put kids physically in the middle of the problem, getting them moving, working together, sharing ideas, and designing solutions" (Cortina, 2015, p. 26). The primary goal of these activities is to "expose students to computing as an intellectual discipline that goes beyond their understanding of computers as a tool and a toy" (p. 26). Cortina writes that these activities are easy to present, because they do not depend on hardware or

software, require few materials, and encourage collaborative work. He adds, “They work well when access to computers is limited or nonexistent” (p. 25).

When students – especially of elementary age – are ready to begin writing their own programs, “Block-based programming is increasingly the way that learners are being introduced to the practice of programming” (Guzdial, 2019, p. 22). Visual programming environments are incredibly supportive of beginners, representing a “low floor” to the practice of programming:

- Blocks can prevent frustrating syntax errors (p. 22);
- The user is provided with a list or toolbox of available commands which can be easily browsed (p. 22);
- Blocks are “conceptually organized and color-coded” (p. 23); and
- Drag-and-drop functionality eliminates the need for typing and finding uncommon characters on the keyboard (p. 23).

Research has shown that block-based tools can welcome learners from historically underrepresented populations to computer science learning, and that students who learned fundamental computer science concepts with blocks tend to score higher on content assessments than students who learned the same concepts in a text-based language (pp. 23-24). It is not necessarily the case, however, that block-based programming significantly aids students in the eventual transition to text-based programming; and as students age, they tend to feel that using blocks does not constitute “real” programming (pp. 24-25). Therefore, if programming is to have a place in a comprehensive technology education scope and sequence, block-based experiences should follow some computational thinking practice, and an eventual transition to text-based languages should be made. The following learning materials are suggested resources to aid teachers in the delivery of content related to the computer science standards:

Resource/Provider	Grades	Description
Code.org Computer Science Fundamentals	K-5	This comprehensive computer science curriculum is divided into six scaffolded courses (A-F) that correlate to each grade level. They are aligned to the CSTA standards as well as cross-curricular standards where appropriate. The courses include unplugged lessons, block-based coding activities, and digital citizenship lessons from Common Sense Education. Two accelerated "Express" courses are available, one for grades K-2 and one for grades 3 and above.
Dash & Dot robots	K-5	These robots can be programmed with iPads, Android devices, and Chromebooks using block-based or other visual programming languages. A sequence of lessons and programming challenges are aligned to the Computer Science Fundamentals curriculum.
Code.org Computer Science Discoveries	6-10	This six-unit curriculum is designed to respond flexibly to the given context: it can be implemented as a yearlong course, two three-unit courses, or three two-unit courses. The lessons are aligned to the CSTA standards as well as cross-curricular standards where appropriate. Each unit focuses on a different aspect of computer science learning: 1. Problem Solving and Computing 2. Web Development (HTML and CSS) 3. Interactive Animations and Games (JavaScript) 4. The Design Process 5. Data and Society 6. Physical Computing (Adafruit Circuit Playground)
AP Computer Science Principles	9-12	This course is built on the Computer Science Principles framework developed by the College Board, and is offered in different forms by several different curriculum providers, such as Code.org, Edhesive, and CodeHS. It introduces students to foundational concepts in computer science and "challenges them to explore how computing and technology can impact the world."
Edhesive Introduction to Computer Science	9-12	This web-based course prepares students for an AP Computer Science course and uses the Python language to teach students how to think computationally and solve complex problems.

Apple Everyone Can Code: Puzzles	4-7	Using the Swift programming language, this course leverages Apple platforms to engage students in computer science and computational thinking. Each chapter is divided into four sections: Learn, Try, Apply, and Connect. The curriculum includes unplugged activities and journaling opportunities.
Apple Everyone Can Code: Adventures	7-12	This curriculum uses the Swift programming language and builds on skills learned in the Puzzles course.
Cyber.org	7-12	This library of courses contains curricula for several different STEM subjects of interest to advancing students, including: • 3D Printing in the Classroom • CompTIA A+ Exam Preparation • Computational Thinking • Computer Science • Cyber Fundamentals • Cyber Literacy I & II • Cyber Science • Cyber Society • Cybersecurity
Kodable	K-5	This platform provides a programming curriculum that begins with block-based programming, but transitions to basic text-based programming by grade 4. It serves as a ramp-up to the CodeHS 6-12 curriculum pathway.
CodeHS	6-12	This platform provides a variety of computer science curricula that is concerned not only with text-based programming, but the impacts of computing, creative computing, virtual reality, and cybersecurity, as well as AP courses and mobile app development.
Google CS First	4-8	This set of lesson activities from Google utilizes the Scratch programming language to deliver and assess computer science and interdisciplinary standards. Activities are either single-day or multi-day.
Tynker, TynkerJr.	2-5, K-2	These apps and websites provide basic visual programming activities that are engaging and self-paced. Some providers include more formal lesson plans to accompany these resources. These are suitable for widespread curricular adoption or as one-off activities for the Hour of Code.
Scratch, ScratchJr.	2-5, K-2	
codeSpark Academy	K-4	
Box Island	1-4	
CSUnplugged.org	K-12	The CSUnplugged website pioneered the “unplugged” movement described above. The websites include all of their lesson materials, with videos and implementation guides.

Teaching Methods. Computer science teaching methods must be as inclusive and engaging as those discussed in the computer literacy domain. A key strategy of computer science education is pair programming, defined by Krauss and Prottsman as "the act of coding with another person by your side […] designating one person as the 'driver' and the other as the 'navigator'" (Krauss & Prottsman, 2017, p. 26). The driver is using the mouse and keyboard, maneuvering through the project and executing the instructions of the navigator, who is concerned with the big picture and ensuring that the code is logical. Pair programming in an educational setting has the benefits of accommodating an entire class with half as many computing machines, forcing all students to think aloud (a metacognitive strategy for evaluating and improving reasoning), and providing a built-in sounding board for ideas and struggles. Krauss and Prottsman also suggest the following teacher behaviors to facilitate computational thinking and computer science learning:

- Start with unplugged activities – they are particularly effective when introducing new concepts (p. 31)
- Encourage movement – prevent students from sitting for long periods of time at a computer by following the 20/20/20 rule: every 20 minutes, get up and stretch while looking at something 20 feet away for 20 seconds (p. 32)
- Foster critical consumption – a key digital citizenship concept of responsibly analyzing sources of information, an exploration of the ways in which computer science can be used to mislead or exacerbate confirmation bias is particularly engaging (p. 33)
- Protect privacy and prevent cyberbullying – for students under age 13, "Monitor and Protect," and for students 13 and older, "Trust but Verify" (pp. 33-34)

- Achieve access – advocate for necessary funding, and select resources that will achieve equity for students of identities traditionally underrepresented in computer science (p. 35)
- Banish anxiety – adopt the philosophy that sees "everyone a teacher, everyone a learner," and encourage the celebration of failure as a First Attempt In Learning (p. 36)

Krauss and Prottsman also share:

Dos and Don'ts of Teaching Computer Science
1. Don't expect to be an expert
2. Do let your class explore
3. Do let your class share
4. Do give kids time to move
5. Do get creative
6. Don't be a bore
7. Do relate computer science to students' lives
8. Don't expect cookie-cutter results
9. Do set students up for success
10. Do treat computer science as an art
11. Do give it a try

(pp. 38-43)

Hazzan, Lapidot, and Ragonis articulate in their *Guide to Teaching Computer Science* the constructivist philosophy that learning is "an active acquisition of ideas and knowledge construction, rather than a passive process," and that because learners must construct their own mental models, learning requires the individual to be active (Hazzan, Lapidot, & Ragonis, 2014). The authors go on to describe a set of pedagogical tools or teaching methods that have been shown through research to activate learning and support computer science learners' construction of understandings.

First, they highlight pedagogical games, claiming that "A well-planned game enables to learn new concepts in an alternative class atmosphere, involves social interaction, introduces a change in the teaching method, and is a kind of activity that all students are good at" (ch. 7.2.1). Next, they recommend the CS Unplugged approach, followed by a discussion of rich tasks,

which they define as "programming exercises that (a) can be solved in a variety of ways […] and (b) can be solved within the duration of one lesson based on learners' current knowledge" (ch. 7.2.3). These tasks, they argue, can focus on any computer science subject, so long as they have the potential to elicit and promote discussion about more abstract computer science ideas, also referred to as "big ideas." Fourth, concept maps are encouraged as a graphical pedagogical tool to allow students to organize and represent knowledge visually. Finally, to promote thinking at higher levels of abstraction, Hazzan et al. argue for the use of classification of objects and phenomena from life, as well as metaphor, as teaching methods in computer science. According to the authors, "This kind of task relies on the constructivist approach […] Due to the explanatory power of analogy, when learners face difficulties in understanding a new concept […] a metaphor may offer a new perspective on the concept and may support its understanding" (ch. 7.2.5-7.2.6). Hazzan et al. also recommend flipped classroom and project-based learning approaches where appropriate.

Scope and Sequence

The following scope and sequence illustrates an alignment of the selected standards with the Kindergarten through grade 9 technology courses at Overton Public School. Each learning target can be delivered and assessed with a combination of learning materials selected from above in accordance with teacher preference and good teaching practice. The document sets the grade level at which each standard should be first introduced, then reinforced, and finally mastered. Next to each standard is indicated whether the content of that standard could be reasonably integrated in other curricular areas, given that they represent knowledge or skills that students might encounter in other subject areas organically if the teacher is incorporating educational technology. This would alleviate some instructional burdens in computing courses.

Overton Public School — Computer Science and Literacy: Scope and Sequence

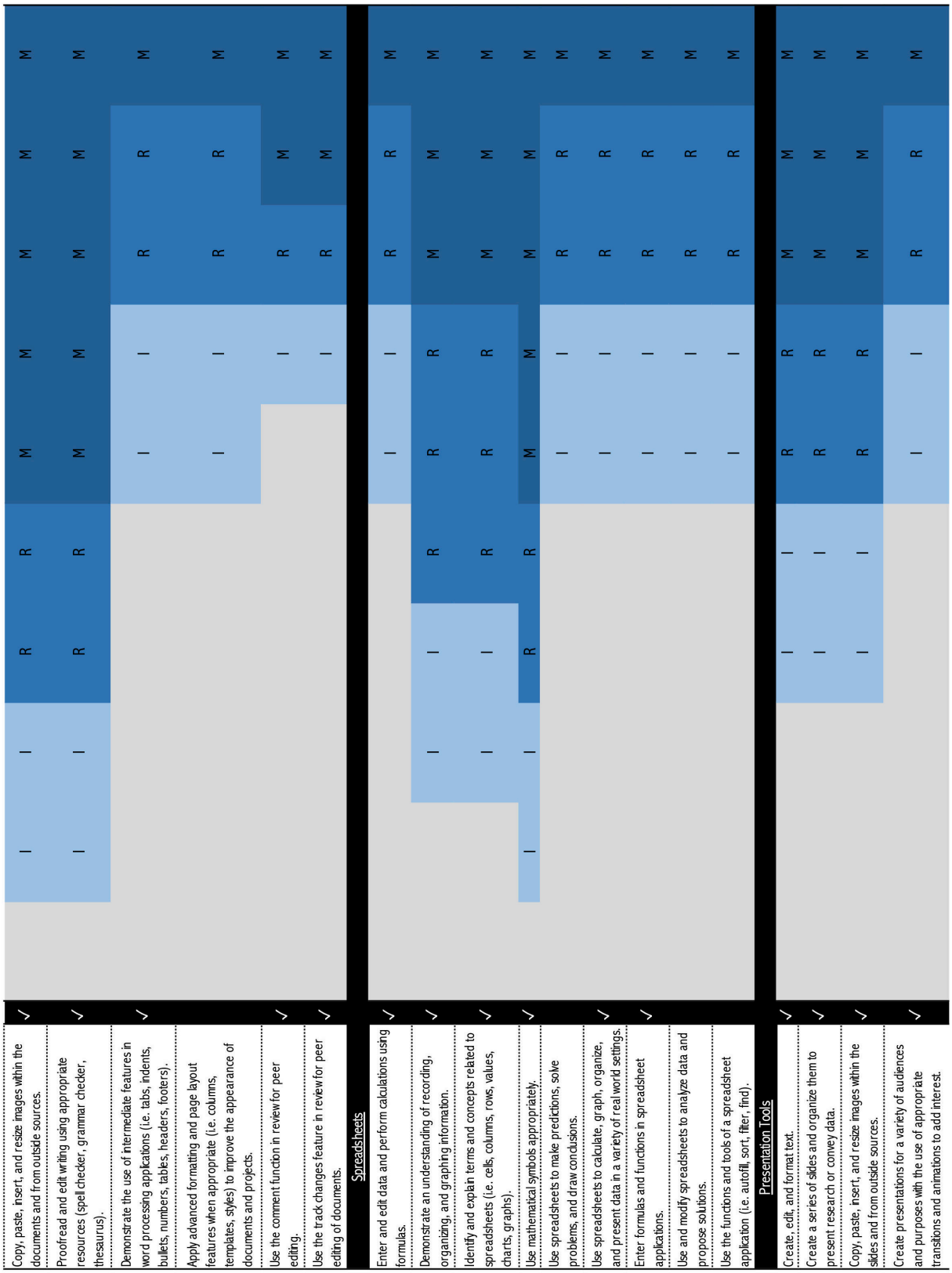
Digital Citizenship

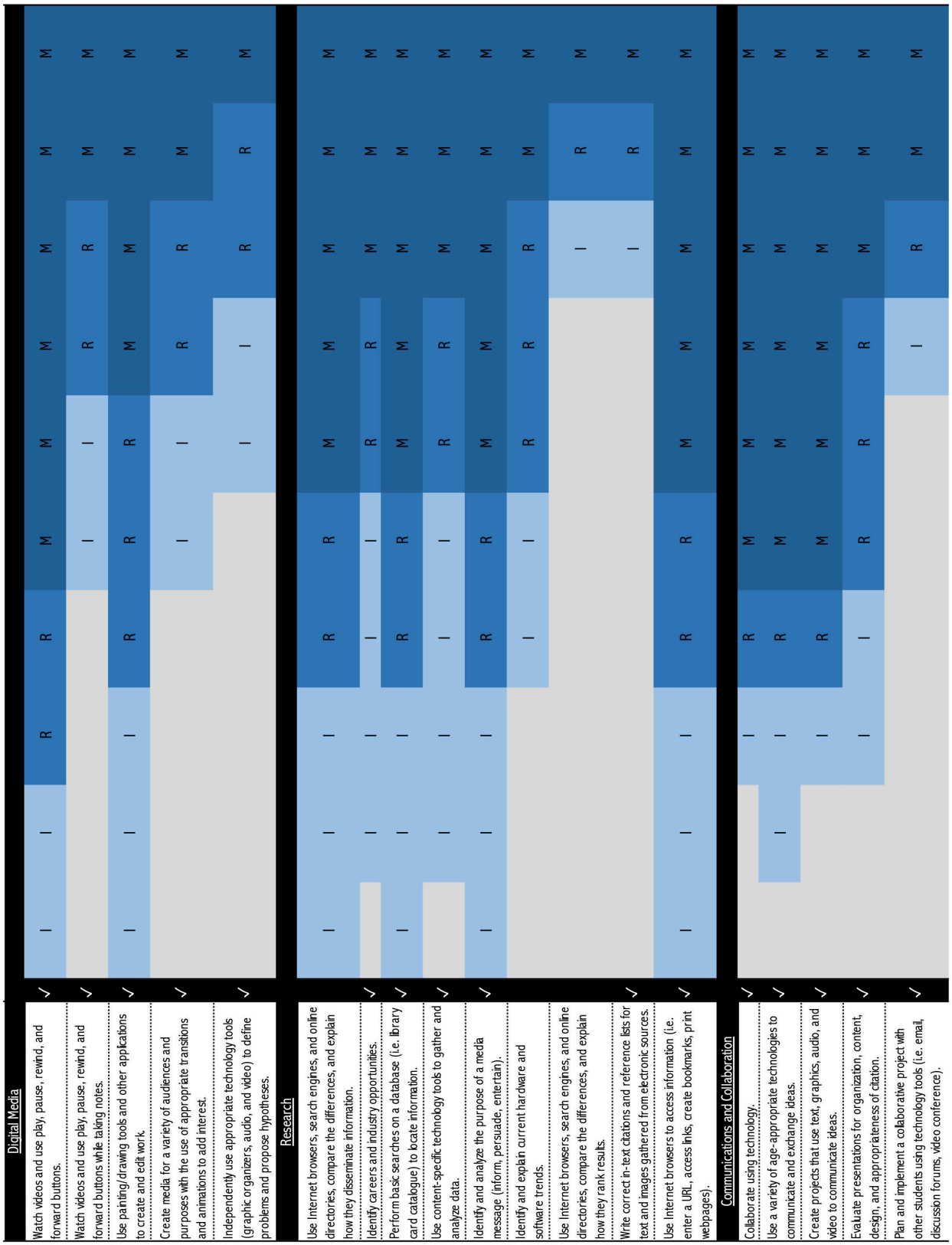
NE K-12 Technology Scope & Sequence	✓	← *Standard may be integrated*							I = Introduce	R = Reinforce	M = Master
		K	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
Responsible Use											
Demonstrate compliance of Responsible Use Policy and classroom rules regarding technology use and networks.	✓	I	I	I	R	R	R	R	M	M	M
Explain responsible uses of technology and digital information and describe potential consequences of inappropriate use.	✓	I	I	I	R	R	R	R	M	M	M
Identify and explain the strategies for the safe and efficient use of computers (passwords, virus protection software, etc.).				I	I	R	R	R	M	M	M
Demonstrate safe email practices and appropriate email etiquette.	✓				I	I	R	R	M	M	M
Identify cyberbullying and describe strategies to deal with such a situation.	✓		I	I	I	R	R	M	M	M	M
Explore social and ethical impacts of technology.	✓	I	I	I	R	R	M	M	M	M	M
Recognize and describe the potential risks and dangers associated with online communication.	✓	I	I	I	R	R	M	M	M	M	M
Give examples of hardware and software that enable people with disabilities to use technology.						I	I	R	R	M	M
Analyze and explain how media and data can be used to distort, exaggerate, and misinterpret information.	✓				I	I	R	R	R	M	M
Explain the potential risks associated with the use of networked digital environments (Internet, cell phones, wireless networks) and sharing personal information.						I	I	R	R	M	M
Copyright											
Explain fair use guidelines for copyrighted material (images, music, videos, etc.)	✓			I	I	R	R	R	M	M	M
ISTE Standards for Students											
Students cultivate and manage their digital identity and are aware of the permanence of their actions in the digital world.				I	I	I	R	R	M	M	M
Students engage in positive, safe, legal, and ethical behavior when using technology including social interactions online or when using networked devices.	✓	I	I	I	R	R	R	R	M	M	M

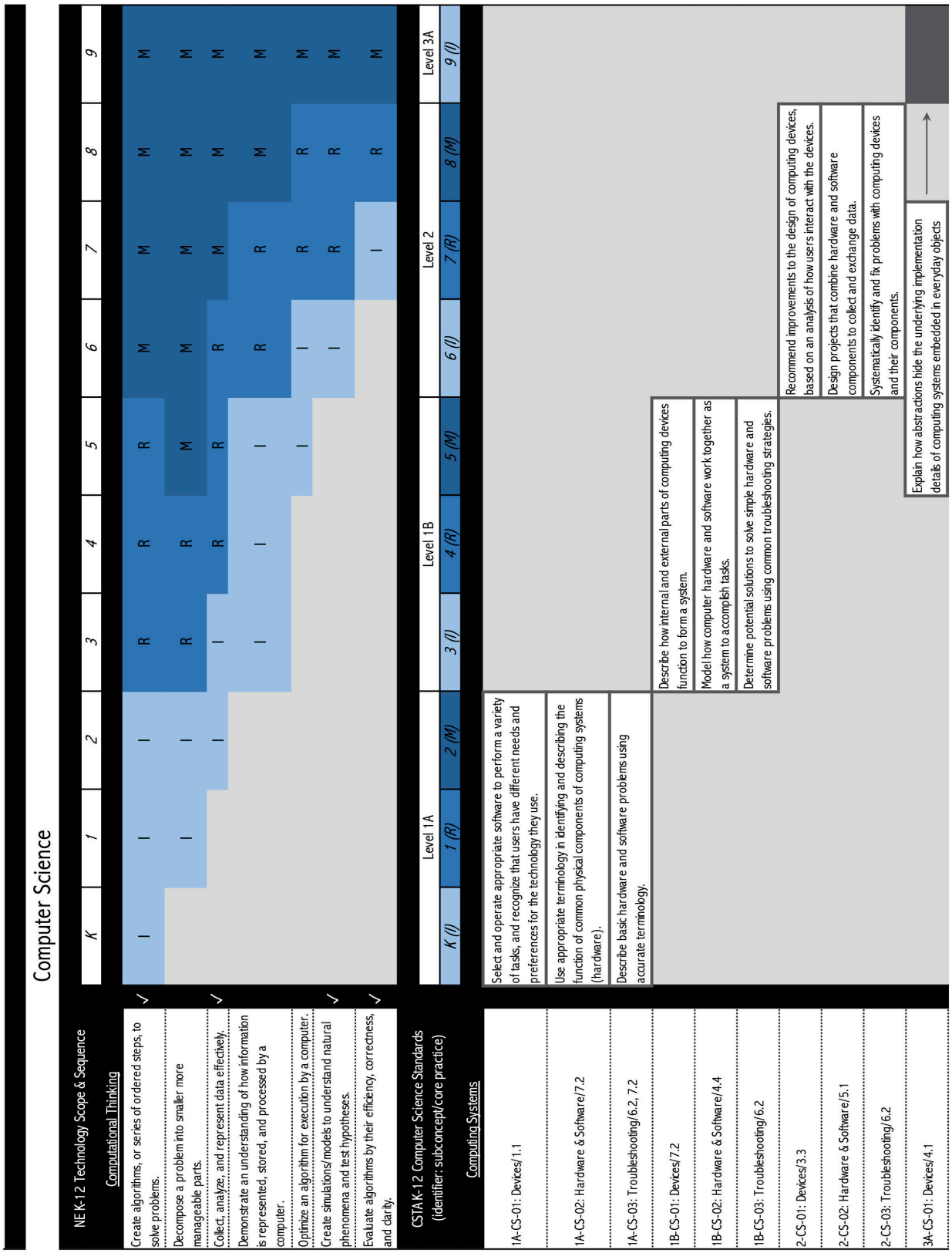
Students demonstrate an understanding of and respect for the rights and obligations of using and sharing intellectual property.	✓			I	I	I	R	R	M	M	M
Students manage their personal data to maintain digital privacy and security and are aware of data-collection technology used to track their navigation online.				I	I	I	R	R	M	M	M

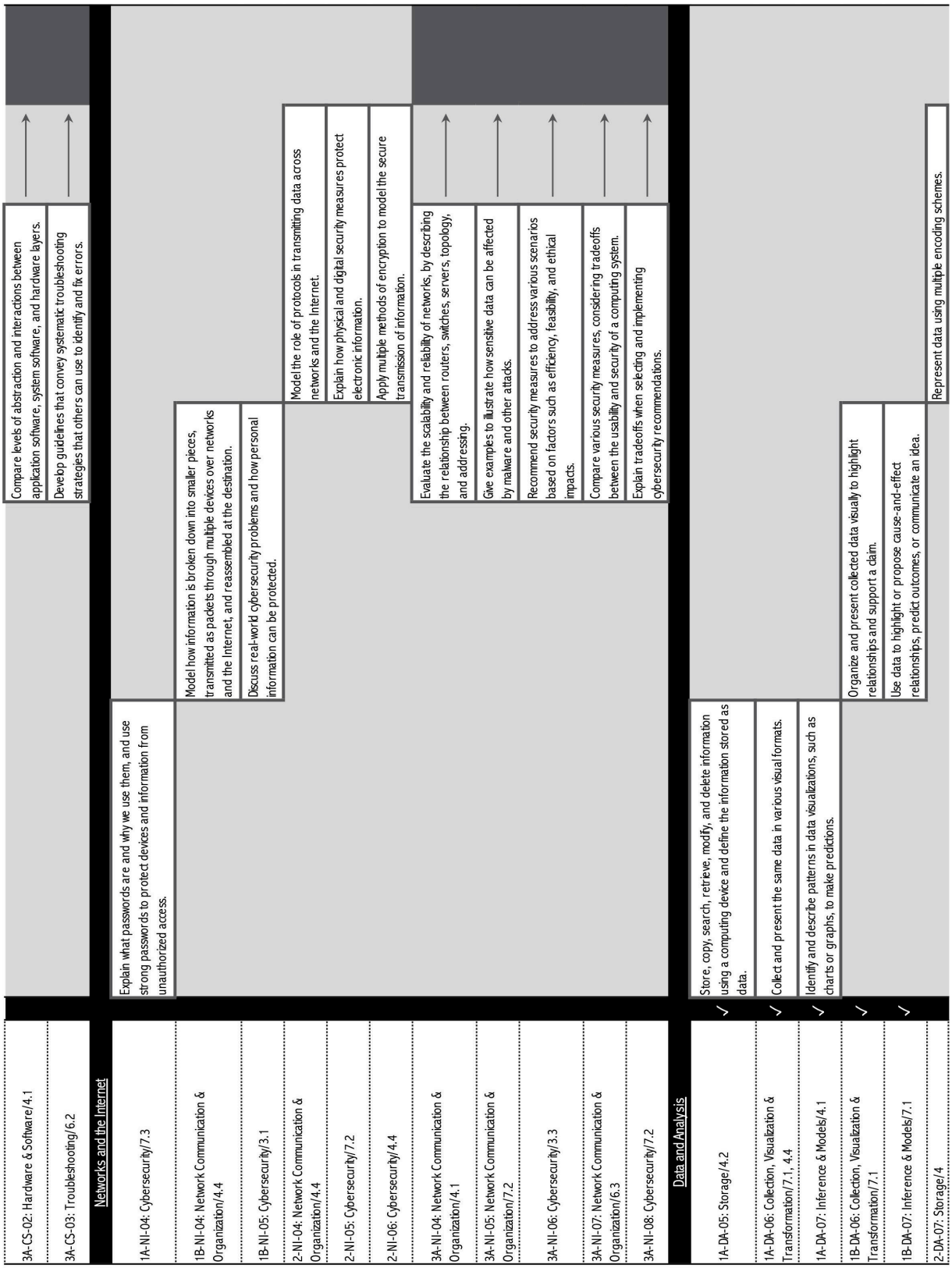
Computer Literacy

NE K-12 Technology Scope & Sequence		K	1	2	3	4	5	6	7	8	9
Keyboarding											
Use proper posture and ergonomics.				I	I	I	R	R	R	M	M
Locate and use letter and number keys with left and right hand placement.				I	I	I	R	R	R	M	M
Locate and use correct finger/hand for spacebar, enter, and shift key.				I	I	I	R	R	R	M	M
Gain proficiency and speed in keyboarding. (Type 5 WPM per grade level beginning at 2nd grade.)				I (5)	I (10)	I (15)	R (20)	R (25)	R (30)	M (35)	M (40)
File Management											
Organize files and folders.	✓			I	I	R	R	M	M	M	M
Manage files and save documents.	✓			I	I	R	R	M	M	M	M
Operate Basic Device Functionality											
Turn on the computer.	✓	I	I	I	R	R	M	M	M	M	M
Login and logoff the computer.	✓	I	I	I	R	R	M	M	M	M	M
Use a pointing device to click menus and icons.	✓	I	I	I	R	R	M	M	M	M	M
Open programs, web apps, and documents.	✓	I	I	I	R	R	M	M	M	M	M
Use buttons and media players.	✓	I	I	I	R	R	M	M	M	M	M
Hardware and Software											
Demonstrate an understanding of the relationship between hardware and software.		I	I	I	R	R	M	M	M	M	M
Troubleshoot basic hardware and software problems.	✓						I	I	R	R	M
Identify major computer components.		I	I	I	R	R	R	R	R	M	M
Describe the components and functions of computers and networks.					I	I	R	R	R	M	M
Apply strategies for identifying and solving routine problems that occur during everyday computer use.	✓	I	I	I	R	R	R	R	R	M	M
Word Processing											
Write, edit, save, and print documents in one sitting.	✓		I	I	R	R	M	M	M	M	M
Use menu/toolbar functions, such as font size, font style, and line spacing to format a document.	✓		I	I	R	R	M	M	M	M	M
Highlight, copy, and paste text.	✓		I	I	R	R	M	M	M	M	M


✓
Copy, paste, insert, and resize images within the documents and from outside sources.
I I R R M M M M M
Proofread and edit writing using appropriate resources (spell checker, grammar checker, thesaurus).
I I R R M M M M M
✓
Demonstrate the use of intermediate features in word processing applications (i.e. tabs, indents, bullets, numbers, tables, headers, footers).
I I R R M
✓
Apply advanced formatting and page layout features when appropriate (i.e. columns, templates, styles) to improve the appearance of documents and projects.
I I R R M
✓
Use the comment function in review for peer editing.
I R M M
✓
Use the track changes feature in review for peer editing of documents.
I R M M
Spreadsheets
✓
Enter and edit data and perform calculations using formulas.
I I R R M
✓
Demonstrate an understanding of recording, organizing, and graphing information.
I I R R R M M M M
✓
Identify and explain terms and concepts related to spreadsheets (i.e. cells, columns, rows, values, charts, graphs).
I I R R R M M M M
✓
Use mathematical symbols appropriately.
I I R R M M M M M
✓
Use spreadsheets to make predictions, solve problems, and draw conclusions.
I I R R M
✓
Use spreadsheets to calculate, graph, organize, and present data in a variety of real world settings.
I I R R M
✓
Enter formulas and functions in spreadsheet applications.
I I R R M
✓
Use and modify spreadsheets to analyze data and propose solutions.
I I R R M
✓
Use the functions and tools of a spreadsheet application (i.e. autofill, sort, filter, find).
I I R R M
Presentation Tools
✓
Create, edit, and format text.
I I R R M M M
✓
Create a series of slides and organize them to present research or convey data.
I I R R M M M
✓
Copy, paste, insert, and resize images within the slides and from outside sources.
I I R R M M M
✓
Create presentations for a variety of audiences and purposes with the use of appropriate transitions and animations to add interest.
I I R R M

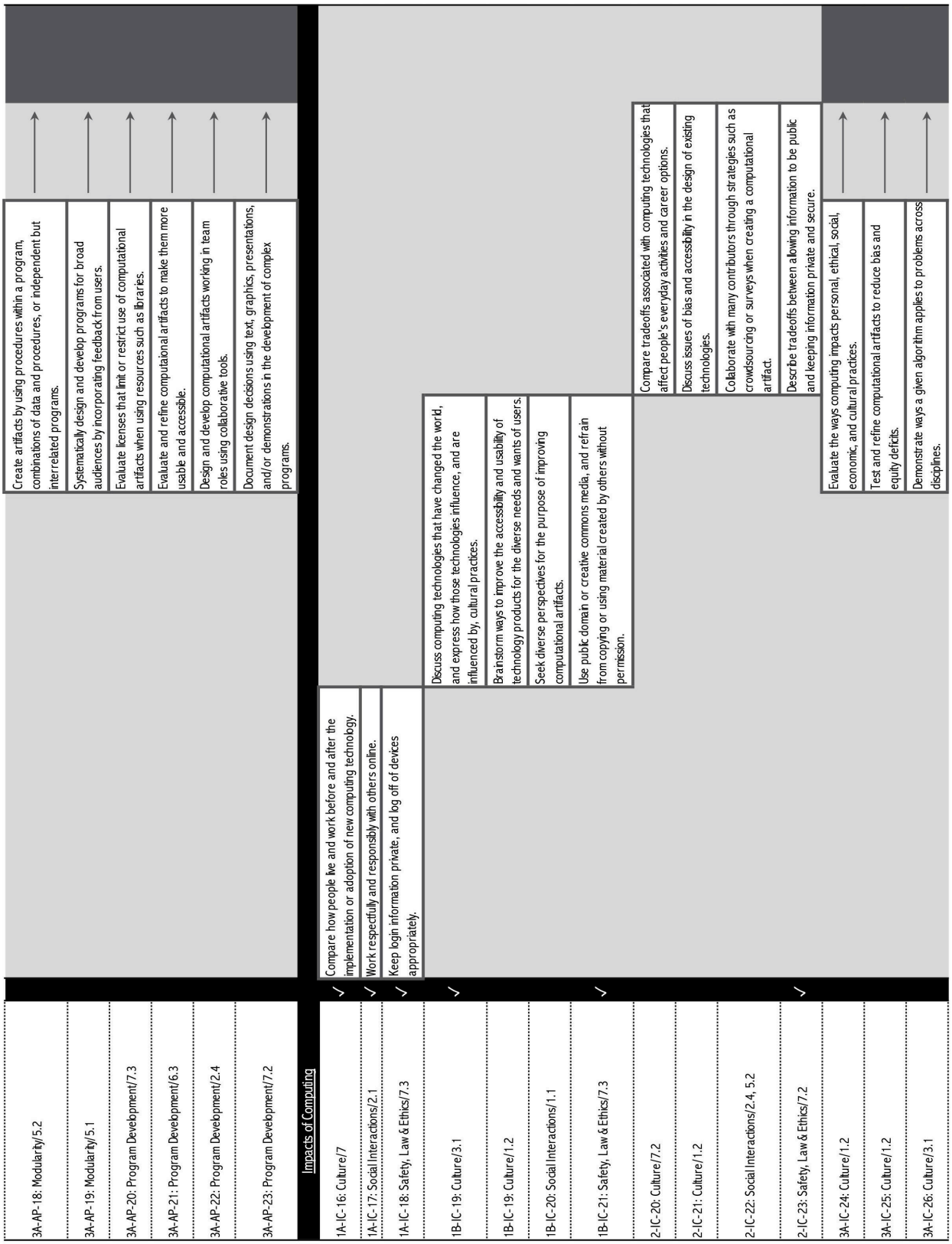

Digital Media
Watch videos and use play, pause, rewind, and forward buttons.
Watch videos and use play, pause, rewind, and forward buttons while taking notes.
Use painting/drawing tools and other applications to create and edit work.
Create media for a variety of audiences and purposes with the use of appropriate transitions and animations to add interest.
Independently use appropriate technology tools (graphic organizers, audio, and video) to define problems and propose hypotheses.
Research
Use Internet browsers, search engines, and online directories, compare the differences, and explain how they disseminate information.
Identify careers and industry opportunities.
Perform basic searches on a database (i.e. library card catalogue) to locate information.
Use content-specific technology tools to gather and analyze data.
Identify and analyze the purpose of a media message (inform, persuade, entertain).
Identify and explain current hardware and software trends.
Use Internet browsers, search engines, and online directories, compare the differences, and explain how they rank results.
Write correct in-text citations and reference lists for text and images gathered from electronic sources.
Use Internet browsers to access information (i.e. enter a URL, access links, create bookmarks, print webpages).
Communications and Collaboration
Collaborate using technology.
Use a variety of age-appropriate technologies to communicate and exchange ideas.
Create projects that use text, graphics, audio, and video to communicate ideas.
Evaluate presentations for organization, content, design, and appropriateness of citation.
Plan and implement a collaborative project with other students using technology tools (i.e. email, discussion forums, video conference).

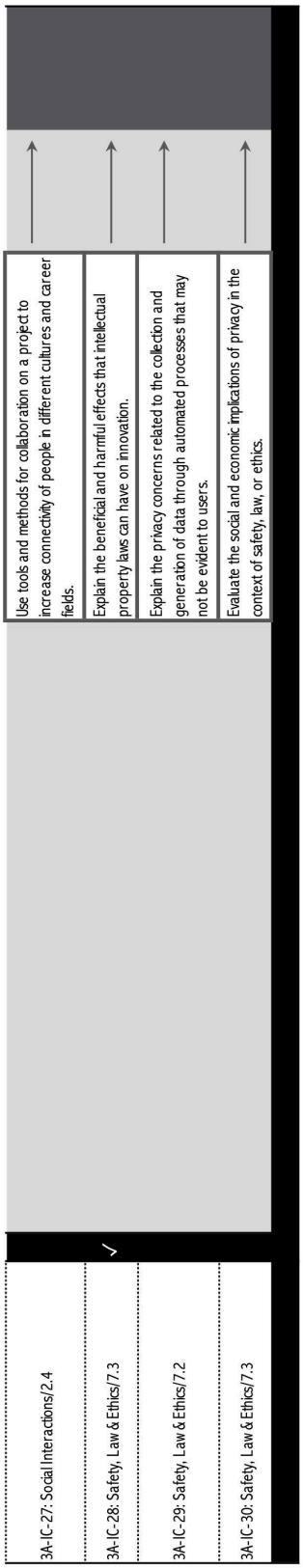

Computer Science
NE K-12 Technology Scope & Sequence
Computational Thinking
Create algorithms, or series of ordered steps, to solve problems.
Decompose a problem into smaller more manageable parts.
Collect, analyze, and represent data effectively.
Demonstrate an understanding of how information is represented, stored, and processed by a computer.
Optimize an algorithm for execution by a computer.
Create simulations/models to understand natural phenomena and test hypotheses.
Evaluate algorithms by their efficiency, correctness, and clarity.
CSTA K-12 Computer Science Standards (identifier: subconcept/core practice)
Computing Systems
1A-CS-01: Devices/1.1
1A-CS-02: Hardware & Software/7.2
1A-CS-03: Troubleshooting/6.2, 7.2
1B-CS-01: Devices/7.2
1B-CS-02: Hardware & Software/4.4
1B-CS-03: Troubleshooting/6.2
2-CS-01: Devices/3.3
2-CS-02: Hardware & Software/5.1
2-CS-03: Troubleshooting/6.2
3A-CS-01: Devices/4.1
K
1
2
3
4
5
6
7
8
9
Level 1A
Level 1B
Level 2
Level 3A
K (I)
1 (R)
2 (M)
3 (I)
4 (R)
5 (M)
6 (I)
7 (R)
8 (M)
9 (I)
Select and operate appropriate software to perform a variety of tasks, and recognize that users have different needs and preferences for the technology they use.
Use appropriate terminology in identifying and describing the function of common physical components of computing systems (hardware).
Describe basic hardware and software problems using accurate terminology.
Describe how internal and external parts of computing devices function to form a system.
Model how computer hardware and software work together as a system to accomplish tasks.
Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies.
Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.
Design projects that combine hardware and software components to collect and exchange data.
Systematically identify and fix problems with computing devices and their components.
Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects


3A-CS-02: Hardware & Software/4.1
3A-CS-03: Troubleshooting/6.2
Networks and the Internet
1A-NI-04: Cybersecurity/7.3
1B-NI-04: Network Communication & Organization/4.4
1B-NI-05: Cybersecurity/3.1
2-NI-04: Network Communication & Organization/4.4
2-NI-05: Cybersecurity/7.2
2-NI-06: Cybersecurity/4.4
3A-NI-04: Network Communication & Organization/4.1
3A-NI-05: Network Communication & Organization/7.2
3A-NI-06: Cybersecurity/3.3
3A-NI-07: Network Communication & Organization/6.3
3A-NI-08: Cybersecurity/7.2
Data and Analysis
1A-DA-05: Storage/4.2
1A-DA-06: Collection, Visualization & Transformation/7.1, 4.4
1A-DA-07: Inference & Models/4.1
1B-DA-06: Collection, Visualization & Transformation/7.1
1B-DA-07: Inference & Models/7.1
2-DA-07: Storage/4
Compare levels of abstraction and interactions between application software, system software, and hardware layers.
Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
Explain what passwords are and why we use them, and use strong passwords to protect devices and information from unauthorized access.
Model how information is broken down into smaller pieces, transmitted as packets through multiple devices over networks and the Internet, and reassembled at the destination.
Discuss real-world cybersecurity problems and how personal information can be protected.
Model the role of protocols in transmitting data across networks and the Internet.
Explain how physical and digital security measures protect electronic information.
Apply multiple methods of encryption to model the secure transmission of information.
Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing.
Give examples to illustrate how sensitive data can be affected by malware and other attacks.
Recommend security measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.
Compare various security measures, considering tradeoffs between the usability and security of a computing system.
Explain tradeoffs when selecting and implementing cybersecurity recommendations.
Store, copy, search, retrieve, modify, and delete information using a computing device and define the information stored as data.
Collect and present the same data in various visual formats.
Identify and describe patterns in data visualizations, such as charts or graphs, to make predictions.
Organize and present collected data visually to highlight relationships and support a claim.
Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea.
Represent data using multiple encoding schemes.

2-DA-08: Collection, Visualization & Transformation/6.3
2-DA-09: Inference & Models/5.3, 4.4
3A-DA-09: Storage/4.1
3A-DA-10: Storage/3.3
3A-DA-11: Collection, Visualization & Transformation/4.4
3A-DA-12: Inference & Models/4.4
Collect data using computational tools and transform the data to make it more useful and reliable.
Refine computational models based on the data they have generated.
Translate between different bit representations of real-world phenomena, such as characters, numbers, and images.
Evaluate the tradeoffs in how data elements are organized and where data is stored.
Create interactive data visualizations using software tools to help others better understand real-world phenomena.
Create computational models that represent the relationships among different elements of data collected from a phenomenon or process.
Algorithms and Programming
1A-AP-08: Algorithms/4.4
1A-AP-09: Variables/4.4
1A-AP-10: Control/5.2
1A-AP-11: Modularity/3.2
1A-AP-12: Program Development/5.1, 7.2
1A-AP-13: Program Development/7.3
1A-AP-14: Program Development/6.2
1A-AP-15: Program Development/7.2
1B-AP-08: Algorithms/6.3, 3.3
1B-AP-09: Variables/5.2
1B-AP-10: Control/5.2
1B-AP-11: Modularity/3.2
1B-AP-12: Modularity/5.3
1B-AP-13: Program Development/1.1, 5.1
Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.
Model the way programs store and manipulate data by using numbers or other symbols to represent information.
Develop programs with sequences and simple loops, to express ideas or address a problem.
Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.
Develop plans that describe a program's sequence of events, goals, and expected outcomes.
Give attribution when using the ideas and creations of others while developing programs.
Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.
Using correct terminology, describe steps taken and choices made during the iterative process of program development.
Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
Create programs that use variables to store and modify data.
Create programs that include sequences, events, loops, and conditionals.
Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.
Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences.

1B-AP-14: Program Development/5.2, 7.3
1B-AP-15: Program Development/6.1, 6.2
1B-AP-16: Program Development/2.2
1B-AP-17: Program Development/7.2
2-AP-10: Algorithms/4.4, 4.1
2-AP-11: Variables/5.1, 5.2
2-AP-12: Control/5.1, 5.2
2-AP-13: Modularity/3.2
2-AP-14: Modularity/4.1, 4.3
2-AP-15: Program Development/2.3, 1.1
2-AP-16: Program Development/4.2, 5.2, 7.3
2-AP-17: Program Development/6.1
2-AP-18: Program Development/2.2
2-AP-19: Program Development/7.2
3A-AP-13: Algorithms/5.2
3A-AP-14: Variables/4.1
3A-AP-15: Control/5.2
3A-AP-16: Control/5.2
3A-AP-17: Control/3.2
Observe intellectual property rights and give appropriate attribution when creating or remixing programs.
Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.
Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development.
Describe choices made during program development using code comments, presentations, and demonstrations.
Use flowcharts and/or pseudocode to address complex problems as algorithms.
Create clearly named variables that represent different data types and perform operations on their values.
Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
Create procedures with parameters to organize code and make it easier to reuse.
Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
Incorporate existing code, media, and libraries into original programs, and give attribution.
Systematically test and refine programs using a range of test cases.
Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
Document programs in order to make them easier to follow, test, and debug.
Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.
Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to iniate instructions.
Decompose problems into smaller components through systemic analysis, using constructs such as procedures, modules, and/or objects.


3A-AP-18: Modularity/5.2
3A-AP-19: Modularity/5.1
3A-AP-20: Program Development/7.3
3A-AP-21: Program Development/6.3
3A-AP-22: Program Development/2.4
3A-AP-23: Program Development/7.2
Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
Systematically design and develop programs for broad audiences by incorporating feedback from users.
Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.
Evaluate and refine computational artifacts to make them more usable and accessible.
Design and develop computational artifacts working in team roles using collaborative tools.
Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.
Impacts of Computing
1A-IC-16: Culture/7
1A-IC-17: Social Interactions/2.1
1A-IC-18: Safety, Law & Ethics/7.3
1B-IC-19: Culture/3.1
1B-IC-19: Culture/1.2
1B-IC-20: Social Interactions/1.1
1B-IC-21: Safety, Law & Ethics/7.3
2-IC-20: Culture/7.2
2-IC-21: Culture/1.2
2-IC-22: Social Interactions/2.4, 5.2
2-IC-23: Safety, Law & Ethics/7.2
3A-IC-24: Culture/1.2
3A-IC-25: Culture/1.2
3A-IC-26: Culture/3.1
Compare how people live and work before and after the implementation or adoption of new computing technology.
Work respectfully and responsibly with others online.
Keep login information private, and log off of devices appropriately.
Discuss computing technologies that have changed the world, and express how those technologies influence, and are influenced by, cultural practices.
Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users.
Seek diverse perspectives for the purpose of improving computational artifacts.
Use public domain or creative commons media, and refrain from copying or using material created by others without permission.
Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.
Discuss issues of bias and accessibility in the design of existing technologies.
Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.
Describe tradeoffs between allowing information to be public and keeping information private and secure.
Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.
Test and refine computational artifacts to reduce bias and equity deficits.
Demonstrate ways a given algorithm applies to problems across disciplines.


3A-IC-27: Social Interactions/2.4
3A-IC-28: Safety, Law & Ethics/7.3
3A-IC-29: Safety, Law & Ethics/7.2
3A-IC-30: Safety, Law & Ethics/7.3
Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields.
Explain the beneficial and harmful effects that intellectual property laws can have on innovation.
Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.
Evaluate the social and economic implications of privacy in the context of safety, law, or ethics.

Conclusion

The primary objective of the technology education curriculum at Overton Public School is to prepare all students for a life after high school which will require them to be technologically-literate, and able to creatively solve problems using efficiency and collaboration. In spite of this worthy goal, the existing program of learning has revealed itself to be inadequate in achieving measurable progress. Exacerbating factors include misalignment in the organization and implementation of learning targets, learning materials, and teaching methods – both among technology faculty and with respect to existing research-based practices. This document serves to remedy curricular deficiencies by rationalizing the need for change through extensive research, negotiating common language and student achievement goals, suggesting learning materials and teaching methods that are rooted in sound pedagogical research, and establishing a customized scope and sequence that leverages all of the above to support positive learning outcomes for all students while respecting the agency of teachers to make the best decisions for their contexts.

It is expected that over time, this document may see revisions that further unify approaches to technology education, reflect changes in state or national learning standards, append new learning materials or teaching methods, or update contextual realities such as course offerings, teaching assignments, enrollment, and more. It has been said that the process of developing and implementing a curriculum is not a "spectator sport"; rather, it demands ongoing preparation and participation from both teachers and administrators. As this work concludes, a new endeavor commences: to willfully and conscientiously apply the presented framework to a new generation of students, recognizing the prescience of a quote likely misattributed to John Dewey, but no less inspiring, "If we teach today's students as we taught yesterday's, we rob them of tomorrow." All students deserve the opportunity to become the innovators of tomorrow.

References

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

Blomstedt, M. L. (2018, December 17). Computer science education in Nebraska. Retrieved from Nebraska Department of Education: cdn.education.ne.gov/wp-content/uploads/2019/05/Computer-Science-Letter.pdf

Buckler, C., Koperski, K., & Loveland, T. R. (2018, December/January). Is computer science compatible with technological literacy? Technology and Engineering Teacher, pp. 15-20.

Cortina, T. J. (2015, March). Broadening participation: Reaching a broader population of students through "unplugged" activities. Communications of the ACM, 58(3), 25-27.

CSTA. (2017). K-12 Computer Science Standards. Retrieved from Computer Science Teachers Association: www.csteachers.org/page/standards

CyberWise. (2020). Our top digital citizenship resources. Retrieved from CyberWise: http://cyberwise.org/digital-citizenship-resources

Domain. (2020). Retrieved from Oxford Online Dictionary: https://en.oxforddictionaries.com/definition/domain

Fluck, A., Webb, M., Cox, M., Charoula, A., Malyn-Smith, J., Voogt, J., & Zagami, J. (2016). Arguing for computer science in the school curriculum. Journal of Educational Technology & Society, 19(3), 38-46.

Gandal, M. (1995). Why we need academic standards. Educational Leadership, 53(1), pp. 84-86.

Guzdial, M. (2019, August). Block-based programming in computer science education. Communications of the ACM, 62(8), 22-25.

Hazzan, O., Lapidot, T., & Ragonis, N. (2014). Guide to teaching computer science: An activity-based approach. London: Springer-Verlag.

Heinlein, L. M., & Shinn, M. (2000). School mobility and student achievement in an urban setting. Psychology in the Schools, 37(4), 349-357.

Immigration Policy Center. (2013). Always in demand: The economic contributions of immigrant scientists and engineers. Washington, D.C.: American Immigration Council.

ISTE. (2016). ISTE standards for students. Retrieved from International Society for Technology in Education: http://iste.org/standards/for-students

K-12 Computer Science Framework. (2016). Retrieved from http://www.k12cs.org

Kindsiko, E., Aidla, A., Poltimäe, H., & Türk, K. (2020). They only teach us word and excel! Trames, 24(1), 53-69.

Krauss, J., & Prottsman, K. (2017). Computational thinking and coding for every student. Thousand Oaks, CA: Corwin.

NCWIT. (2016). Engagement practices framework. Retrieved from National Center for Women & Information Technology: http://ncwit.org/engagement-practices-framework

NDE. (2018). Nebraska K-12 Technology Scope & Sequence. Retrieved from Nebraska Department of Education: www.education.ne.gov/nce/cis

Ribble, M. (2015). Digital citizenship in schools: Nine elements all students should know. Eugene, OR: International Society for Technology in Education.

Simons, P. R. (1999). Transfer of learning: Paradoxes for learners. International Journal of Educational Research, 31(7), 577-589.

Wilson, C. (2013, November). Making computer science count. Communications of the ACM, 56(11), 32-33.