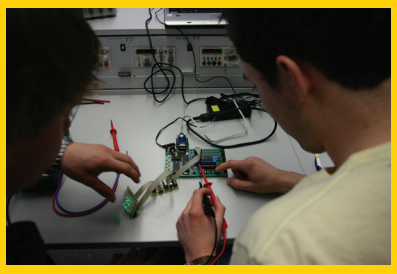
• RELEVANCE • RELATIONSHIPS • RIGOR • RELEVANCE • RELATIONSHIPS • RIGOR • RELEVA
ATIONSHIPS • RIGOR • RELEVANCE • RELATIONSHIPS • RIGOR • RELEVANCE • RELATIONSHI
R • RELEVANCE • RELATIONSHIPS • RIGOR • RELEVANCE • RELATIONSHIPS • RIGOR • RELEVA
ATIONSHIPS • RIGOR • RELEVANCE • RELATIONSHIPS • RIGOR • RELEVANCE • RELATIONSHI
R • RELEVANCE • RELATIONSHIPS • RIGOR • RELEVANCE • RELATIONSHIPS • RIGOR • RELEVA
ATIONSHIPS • RIGOR • RELEVANCE • RELATIONSHIPS • RIGOR • RELEVANCE • RELATIONSH



Michigan K-12 Standards

Computer Science

May 2019

MICHIGAN Department of Education

CONTENTS

INTRODUCTION

The Michigan Computer Science Standards are the result of a variety of efforts focused on preparing Michigan's learners for a world that demands much more of them than of students of previous generations. The demand for computer science education is high. According to a Google & Gallup survey (2015), a high percentage of parents perceive computer science careers positively and would like their school to offer computer science. Research from Horizon Media (2015) found that 86% of Americans believe computer science should be considered a "basic skill" (source: PRN Newswire October 5, 2015). In response, this document outlines Michigan's effort to address the need for students to acquire computer science knowledge and skills. It includes a recommended set of Computer Science Standards for Michigan students in Grades K-12.

CONNECTION TO TOP 10 IN 10 AND OTHER MICHIGAN EFFORTS

Over the past several years, a collaborative effort among educators, government, and private sector partners has been shaping the direction of K-12 education to equip Michigan students. The work of transforming learning began by shaping a crystal-clear vision of the possibilities and the conditions necessary to move from vision to reality. The tenets of Michigan's Top 10 in Ten Years provide the rationale for a statewide commitment to the vision.

> *No matter where students live, they should have access to the same high-quality educational opportunities as any other student in Michigan. Likewise, teachers across the state should have equitable opportunities to learn and excel in their profession no matter where they live and teach.*

Toward this end, *MI Roadmap*, Michigan's educational technology plan was developed to direct resources and create new models of learning wherein students:

- use technology and tools strategically in learning and communicating;
- use argument and reasoning to do research to construct arguments and critique the reasoning of others;
- communicate and collaborate effectively with a variety of audiences; and
- solve problems, construct explanations and design solutions.

Michigan Department of Education (2016).

These key competencies identified in *MI Roadmap* set the stage for development of Michigan's Integrated Technology Competencies for Students (MITECS), a set of competencies focused on an integrated approach to learning enhanced by technology rather than stand-alone skills or technology tools. The MITECS help all students prepare for a digital world. Students learn how to leverage technology to communicate creatively, collaborate locally and globally, and model digital citizenship in an increasingly interconnected world. Through the MITECS, students also learn to discern the credibility and relevance of information as they curate and construct knowledge. All efforts focus on students taking an active role in determining personal learning goals, and utilizing technology to demonstrate learning in a variety of ways. Students use design processes to identify problems and create imaginative solutions, becoming problem-solvers who think divergently, learn to take creative risks, and make meaningful connections among ideas through their own exploration.

The deepest connection between MITECs and Michigan's Computer Science Standards is in developing students who are computational thinkers. Computational thinking can be thought of as a set of overlapping problem-solving skills applied in a variety of different settings. While computational thinking is a critical component of computer science, the inclusion of computational thinking across the content areas does not replace or compete with computer science efforts – it complements them.

> *Where computer science is not yet offered, integrating computational thinking into existing disciplines can empower educators and students to better understand and participate in a computational world. And schools already teaching coding and computer science will benefit from weaving computational thinking across disciplines in order to enrich and amplify lessons that are beyond the reach of computer science classes.*

Computational Thinking for a Computational World, p. 24.

Access to computer science opportunities for all Michigan learners is critical to ensuring all excel in learning and beyond.

ALIGNMENT WITH NATIONAL EFFORTS

Nationally, efforts to equip students include updating and expanding Science, Technology, Engineering, and Mathematics (STEM) for All. In the past, national work has focused on improving STEM instruction, supporting active learning, expanding access to rigorous STEM courses, addressing bias, and expanding opportunities for underrepresented students in STEM. The 2017 federal budget included a Computer Science for All plan and allocated funds to school districts to execute ambitious computer science expansion efforts for all students – with a focus on serving those traditionally underrepresented – and to serve as models for replication nationally (*STEM for All*, 2016 White House Blog). In 2018, a group of state and national leaders convened in Washington, DC. to refresh the national STEM plan. Computer science is called out specifically as of paramount importance to America's future workforce. A key finding of the convening is that computer science principles must be integrated across the educational experience. Michigan is aggressively supporting the creation of STEM learning opportunities to equip students. The Computer Science Standards are crucial to this effort.

URGENCY AND EQUITY

Across the country, and specifically in Michigan, there is an urgency to prepare students with strong computer science skills that equip them to navigate new and emerging innovations. "The ubiquity of personal computing and our increasing reliance on technology have changed the fabric of society and day-to-day life" (K12 CS Framework, 2016). Computer science is identified among the top 50 growth areas in our state, while Michigan employers have identified workforce gaps with thousands of computer science jobs going unfilled. A computer science education can lead to a high tech, high-wage sustainable career, with available computing jobs in Michigan growing at three times the average growth rate (Sawyer, presentation May 2018).

At the same time, students lack access to opportunities for learning the very skills in demand. Access gaps have been identified in several areas including gender, geography, and socioeconomics. In adopting the Computer Science Standards, the Michigan Department of Education and schools across the state are committing to Michigan learners that they will be provided equitable access to computer science opportunities regardless of where they live in the state, or the demographics of the schools they attend. The K12 CS Framework, a foundational document for Michigan's Computer Science Standards, "provides a unifying vision to guide computer science from a subject for the fortunate few to an opportunity for all" (p. 3). It addresses what "equitable access" looks like:

> When equity exists, there are appropriate supports based on individual students' needs so that all have the opportunity to achieve similar levels of success…The result of equity is a diverse classroom of students, based on factors such as race, gender, disability, socioeconomic status, and English language proficiency, all of whom have high expectations and feel empowered to learn (p. 23).

VISION

Michigan's vision for computer science education is that all learners will develop foundational computer science skills to solve problems and be constructive citizens. In doing so, students will:

- Learn new approaches to problem solving;
- Harness the power of computational thinking; and
- Use computer science tools to create technology.

PROCESS AND IMPLEMENTATION TIMELINE

The Computer Science Standards project was implemented by the Michigan Department of Education in May 2018. A stakeholder group convened consisting of 50 members representing educators from K-12 and higher education, state government, professional organizations, business and industry. Governed by guiding principles, target timelines were established shaping the development process through monthly meetings as reflected in Figure 1.

Figure 1. Timeline

The timeline allowed for broad stakeholder involvement and input in the standards adoption process. Public input was also sought during the review period from January 20 to March 6, 2019. The final draft was adopted by the Michigan State Board of Education in May 2019.

GUIDING PRINCIPLES

The stakeholder group tasked with developing Michigan's Computer Science standards framed a set of Guiding Principles to serve as a basis for reasoning, and a guide to help prioritize decisions and recommendations. The guidelines identify what is important to Michigan students:

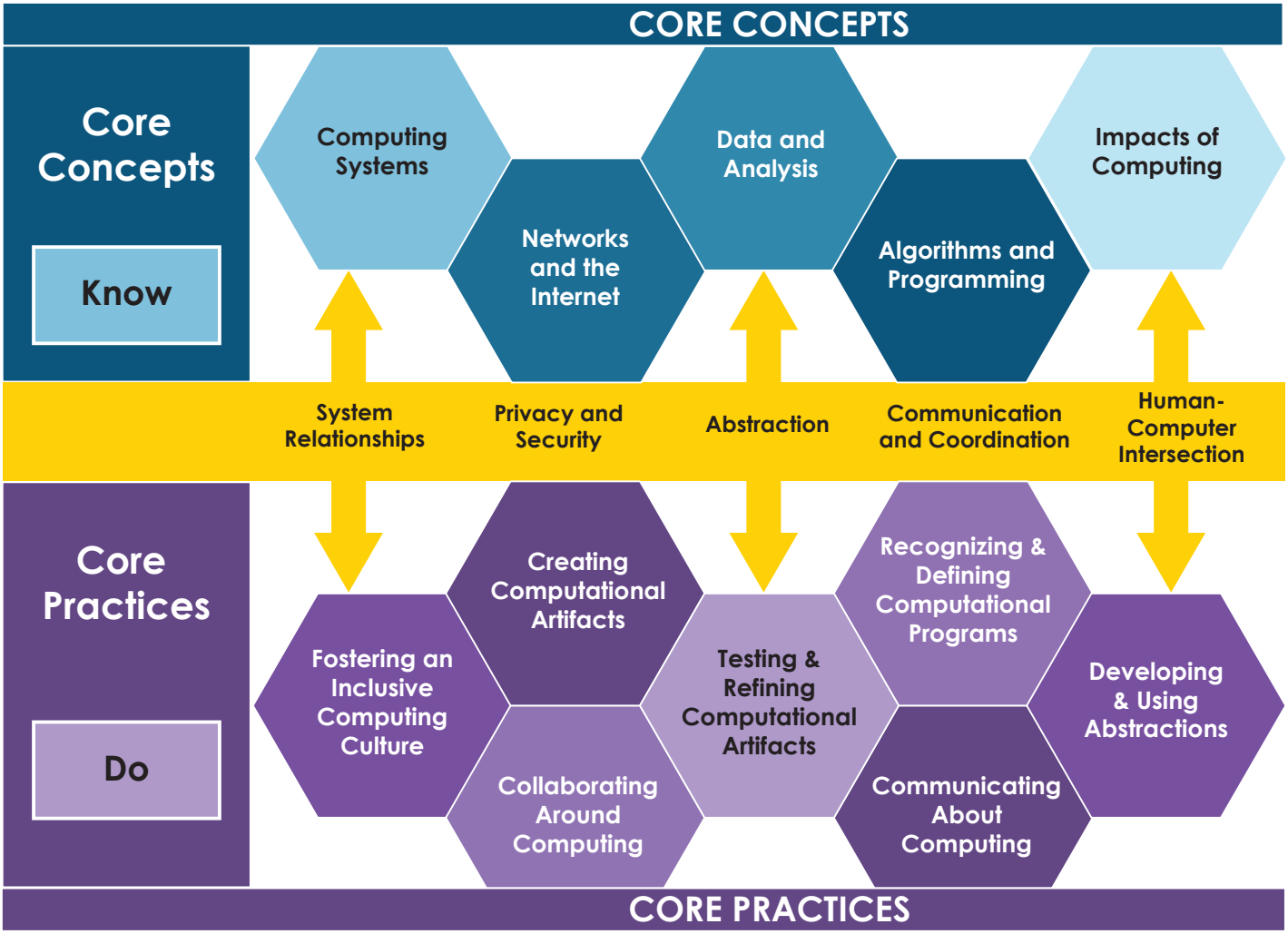
1. Ensure that all students and teachers have equitable access to and participation in computer science.
2. Focus on essential standards that allow for expansion within context.
3. Use research and best practice to drive development and implementation.
4. Align to nationally-recognized standards and frameworks.
5. Enable teachers to implement the curriculum in ways that engage and inspire students and support learning.

STANDARDS BACKGROUND AND ORGANIZATION

Code is all around us. From social apps that keep us connected, to the systems that make a car run, coding touches every sector. Whether they want to grow up to be artists, engineers, scientists, or chefs, software will play a larger role in students' lives than it does today. In a world powered by code, understanding key software concepts has become a new literacy (Seitz, presentation May 2018). The goal in Michigan is support of a system wherein computer science standards are integrated into existing content areas in Kindergarten through Grade 5; while in middle school, students explore courses that develop skills and expertise that continue in early high school, leading to specialization in Grades 11 and 12 as personal interests dictate.

Computing education in K–12 schools includes computer literacy, educational technology, digital citizenship, information technology, and computer science. Computer science digs deeper with "the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society" (Tucker et. al, 2006, in K12 CS Framework). When the Michigan Computer Science Standards stakeholder group began the process of considering the need for standards in Michigan, it studied the K-12 Computer Science Framework (k12cs.org) developed by a cross-sector team convened for similar purpose. The K12 CS Framework "promotes a vision in which all students critically engage in computer science issues; approach problems in innovative ways; and create computational artifacts with a practical, personal, or societal intent" (p. 2). The CS Framework identifies key concepts – what students will know and understand – and practices that reflect what students can do with their learning. These are based on a vision for students who are "not just computer users but also computationally literate creators who are proficient in the concepts and practices of computer science" (p. 4). The core concepts and practices are reflected in Figure 2 below.

Figure 2. K-12 CS Framework Concepts and Practices



The CS Framework has been taken up by many states across the nation as a reliable, representative compilation of the concepts and practices encompassed by the computer science field. The framework is endorsed by leading industries and organizations across the country. After reviewing the K-12 CS Framework and talking with national experts involved in its development, the Michigan stakeholders group determined that the CS Framework would serve as a foundation for the Michigan K-12 Computer Science Standards.

Built upon the K-12 Computer Science Framework, a set of standards were created by the Computer Science Teachers Association (CSTA), which have also served as a model for adoption by other states. The CSTA Standards are licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 40 International (CC BY-NC-SA 4.0) license (https://www.csteachers.org/page/standards). After studying models from other states, engaging in conversation among the experts in computer science, K-12 and higher education, government, business, and industry, the Michigan K-12 CS Standards stakeholder group unanimously recommended adoption of the CSTA Standards for Michigan.

The CSTA Standards:

- Introduce the fundamental concepts of computer science to all students, beginning at the elementary school level.
- Present the computer science in high school in a way that allows districts to deliver course content within the Michigan Merit Curriculum course schema (i.e., computer science content in a fourth year math experience or third year science once required courses have been completed) or as elective credits.
- Encourage schools to offer additional secondary-level computer science courses that will allow

interested students to study facets of computer science in more depth and prepare them for entry into the workforce or college.

- Increase the availability of rigorous computer science for all students, especially those who are members of underrepresented groups. (p. 2)

The standards are written by educators to be coherent and comprehensible to teachers, administrators, and policy makers.

STANDARDS DESIGN

The Michigan Computer Science Standards represent realistic expectations for all students. They identify the computer science knowledge, practices, and skills all students should know and be able to do at each level of their education. The standards are designed to inform and drive a sustainable computer science education program, and may be used as reference points for planning and teaching.

The CS Standards work group recommended adoption of grade bands reflecting developmentally-appropriate standards articulated in the CSTA K-12 Standards document (https://www.csteachers.org/page/standards) that may be implemented in flexible learning environments reflective of students' needs. The recommended bands include:

Level 1A: Lower Elementary (Grades K-2)

Level 1B: Upper Elementary (Grades 3-5)

Level 2: Middle School (Grades 6-8)

Level 3A: High School (Grades 9-10)

Level 3B: High School - Specializing (Grades 11-12)

SECTION LABELING / CODING

Levels 1A, 1B, 2, and 3A are the computer science standards for ALL students. The Level 3B standards are intended for students who wish to pursue the study of computer science in high school beyond what is required for all students (specialty or elective courses).

Coding for each section references back to the Concepts and Practices of the K-12 CS Framework and is illustrated below:

Identifier	Standard	Subconcept	Practice
1A-CS-01	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.	Devices	1.1
1A-CS-02	Use appropriate terminology in identifying and describing the function of common physical components of computing systems (hardware).	Hardware & Software	7.2
1A-CS-03	Describe basic hardware and software problems using accurate terminology.	Troubleshooting	6.2, 7.2

References to the sub-practice descriptors are referred to after the decimal in each practice, i.e., 7.**2**. The table below identifies the practice and its corresponding sub-practice. For a full description, see documents shared on the Resources page.

Practice	Subpractice 1	Subpractice 2	Subpractice 3	Subpractice 4
1. Fostering an Inclusive Computing Culture	Include the unique perspectives of others	Address the needs of diverse end users	Employ self- and peer-advocacy	
2. Collaborating Around Computing	Cultivate working relationships	Create team norms, expectations, and equitable workloads	Solicit and incorporate feedback	Evaluate and select technological tools

Practice	Subpractice 1	Subpractice 2	Subpractice 3	Subpractice 4
3. Recognizing and Defining Computational Problems	Identify complex, interdisciplinary, real-world problems	Decompose complex real-world problems	Evaluate whether it is appropriate and feasible	
4. Developing and Using Abstractions	Extract common features	Evaluate existing technological functionalities and incorporate them	Create modules and develop points of interaction	Model phenomena and processes and simulate systems
5. Creating Computational Artifacts	Plan the development	Create a computational artifact	Modify an existing artifact	
6. Testing and Refining Computational Artifacts	Systematically test	Identify and fix errors	Evaluate and refine	
7. Communicating About Computing	Select, organize, and interpret	Describe, justify, and document	Articulate ideas responsibly	

The K–12 Computer Science Framework, led by the Association for Computing Machinery, Code.org, Computer Science Teachers Association, Cyber Innovation Center, and National Math and Science Initiative in partnership with states and districts, informed the development of this work.

LEVEL 1A: LOWER ELEMENTARY (GRADES K-2)

COMPUTING SYSTEMS

1A-CS-01 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.
Subconcept: Devices; Practice 1.1

1A-CS-02 Use appropriate terminology in identifying and describing the function of common physical components of computing systems (hardware).
Subconcept: Hardware & Software; Practice 7.2

1A-CS-03 Describe basic hardware and software problems using accurate terminology.
Subconcept: Troubleshooting; Practice 6.2, 7.2

NETWORKS AND THE INTERNET

1A-NI-04 Explain what passwords are and why we use them, and use strong passwords to protect devices and information from unauthorized access.
Subconcept: Cybersecurity; Practice 7.3

DATA AND ANALYSIS

1A-DA-05 Store, copy, search, retrieve, modify, and delete information using a computing device and define the information stored as data.
Subconcept: Storage; Practice 4.2

1A-DA-06 Collect and present the same data in various visual formats.
Subconcept: Collection, Visualization & Transformation; Practice 7.1, 4.4

1A-DA-07 Identify and describe patterns in data visualizations, such as charts or graphs, to make predictions.
Subconcept: Inference & Models; Practice 4.1

ALGORITHMS AND PROGRAMMING

1A-AP-08 Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.
Subconcept: Algorithms; Practice 4.4

1A-AP-09 Model the way programs store and manipulate data by using numbers or other symbols to represent information.
Subconcept: Variables; Practice 4.4

1A-AP-10 Develop programs with sequences and simple loops, to express ideas or address a problem.
Subconcept: Control; Practice 5.2

1A-AP-11 Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.
Subconcept: Modularity; Practice 3.2

1A-AP-12 Develop plans that describe a programís sequence of events, goals, and expected outcomes.
Subconcept: Program Development; Practice 5.1, 7.2

1A-AP-13 Give attribution when using the ideas and creations of others while developing programs
Subconcept: Program Development; Practice 7.3

1A-AP-14 Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.
Subconcept: Program Development; Practice 6.2

1A-AP-15 Using correct terminology, describe steps taken and choices made during the iterative process of program development.
Subconcept: Program Development; Practice 7.2

IMPACTS OF COMPUTING

1A-IC-16 Compare how people live and work before and after the implementation or adoption of new computing technology.
Subconcept: Culture; Practice 7

1A-IC-17 Work respectfully and responsibly with others online.
Subconcept: Social Interactions; Practice 2.1

1A-IC-18 Keep login information private, and log off of devices appropriately.
Subconcept: Safety Law & Ethics; Practice 7.3

LEVEL 1B: UPPER ELEMENTARY (GRADES 3-5)

COMPUTING SYSTEMS

1B-CS-01 Describe how internal and external parts of computing devices function to form a system.
Subconcept: Devices*; Practice* 7.2

1B-CS-02 Model how computer hardware and software work together as a system to accomplish tasks.
Subconcept: Hardware & Software*; Practice* 4.4

1B-CS-03 Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies.
Subconcept: Troubleshooting; Practice 6.2

NETWORKS AND THE INTERNET

1B-NI-04 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.
Subconcept: Network Communication & Organization*; Practice* 4.4

1B-NI-05 Discuss real-world cybersecurity problems and how personal information can be protected.
Subconcept: Cybersecurity; Practice 3.1

DATA AND ANALYSIS

1B-DA-06 Organize and present collected data visually to highlight relationships and support a claim.
Subconcept: Collection, Visualization & Transformation; Practice 7.1

1B-DA-07 Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea.
Subconcept: Inference & Models; Practice 7.1

ALGORITHMS AND PROGRAMMING

1B-AP-08 Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
Subconcept: Algorithms; Practice 6.3, 3.3

1B-AP-09 Create programs that use variables to store and modify data.
Subconcept: Variables; Practice 5.2

1B-AP-10 Create programs that include sequences, events, loops, and conditionals.
Subconcept: Control; Practice 5.2

1B-AP-11 Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
Subconcept: Modularity; Practice 3.2

1B-AP-12 Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features.
Subconcept: Modularity; Practice 5.3

1B-AP-13 Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences.
Subconcept: Program Development; Practice 1.1, 5.1

1B-AP-14 Observe intellectual property rights and give appropriate attribution when creating or remixing programs.
Subconcept: Program Development; Practice 5.2, 7.3

1B-AP-15 Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.
Subconcept: Program Development; Practice 6.1, 6.2

1B-AP-16 Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development.
Subconcept: Program Development; Practice 2.2

1B-AP-17 Describe choices made during program development using code comments, presentations, and demonstrations.
Subconcept: Program Development; Practice 7.2

IMPACTS OF COMPUTING

1B-IC-18 Discuss computing technologies that have changed the world, and express how those technologies influence, and are influenced by, cultural practices.
Subconcept: Culture; Practice 3.1

1B-IC-19 Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users.
Subconcept: Culture; Practice 1.2

1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts
Subconcept: Social Interactions; Practice 1.1

1B-IC-21 Use public domain or creative commons media, and refrain from copying or using material created by others without permission.
Subconcept: Safety Law & Ethics; Practice 7.3

LEVEL 2: MIDDLE SCHOOL (GRADES 6-8)

COMPUTING SYSTEMS

2-CS-01 Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.
Subconcept: Devices; Practice 3.3

2-CS-02 Design projects that combine hardware and software components to collect and exchange data.
Subconcept: Hardware & Software; Practice 5.1

2-CS-03 Systematically identify and fix problems with computing devices and their components.
Subconcept: Troubleshooting; Practice 6.2

NETWORKS AND THE INTERNET

2-NI-04 Model the role of protocols in transmitting data across networks and the Internet.
Subconcept: Network Communication & Organization; Practice 4.4

2-NI-05 Explain how physical and digital security measures protect electronic information.
Subconcept: Cybersecurity; Practice 7.2

2-NI-06 Apply multiple methods of encryption to model the secure transmission of information.
Subconcept: Cybersecurity; Practice 4.4

DATA AND ANALYSIS

2-DA-07 Represent data using multiple encoding schemes.
Subconcept: Storage; Practice 4

2-DA-08 Collect data using computational tools and transform the data to make it more useful and reliable.
Subconcept: Collection, Visualization & Transformation; Practice 6.3

2-DA-09 Refine computational models based on the data they have generated.
Subconcept: Inference & Models; Practice 5.3, 4.4

ALGORITHMS AND PROGRAMMING

2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms.
Subconcept: Algorithms; Practice 4.4, 4.1

2-AP-11 Create clearly named variables that represent different data types and perform operations on their values.
Subconcept: Variables; Practice 5.1, 5.2

2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
Subconcept: Control; Practice 5.1, 5.2

2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
Subconcept: Modularity; Practice 3.2

2-AP-14 Create procedures with parameters to organize code and make it easier to reuse.
Subconcept: Modularity; Practice 4.1, 4.3

2-AP-15 Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
Subconcept: Program Development; Practice 2.3, 1.1

2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution.
Subconcept: Program Development; Practice 4.2, 5.2, 7.3

2-AP-17 Systematically test and refine programs using a range of test cases.
Subconcept: Program Development; Practice 6.1

2-AP-18 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
Subconcept: Program Development; Practice 2.2

2-AP-19 Document programs in order to make them easier to follow, test, and debug.
Subconcept: Program Development; Practice 7.2

IMPACTS OF COMPUTING

2-IC-20 Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.
Subconcept: Culture; Practice 7.2

2-IC-21 Discuss issues of bias and accessibility in the design of existing technologies.
Subconcept: Culture; Practice 1.2

2-IC-22 Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.
Subconcept: Social Interactions; Practice 2.4, 5.2

2-IC-23 Describe tradeoffs between allowing information to be public and keeping information private and secure.
Subconcept: Safety Law & Ethics; Practice 7.2

LEVEL 3A: HIGH SCHOOL (GRADES 9-10)

COMPUTING SYSTEMS

3A-CS-01 Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.
Subconcept: Devices; Practice 4.1

3A-CS-02 Compare levels of abstraction and interactions between application software, system software, and hardware layers.
Subconcept: Hardware & Software; Practice 4.1

3A-CS-03 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
Subconcept: Troubleshooting; Practice 6.2

NETWORKS AND THE INTERNET

3A-NI-04 Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing.
Subconcept: Network Communication & Organization; Practice 4.1

3A-NI-05 Give examples to illustrate how sensitive data can be affected by malware and other attacks.
Subconcept: Network Communication & Organization; Practice 7.2

3A-NI-06 Recommend security measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.
Subconcept: Cybersecurity; Practice 3.3

3A-NI-07 Compare various security measures, considering tradeoffs between the usability and security of a computing system.
Subconcept: Network Communication & Organization; Practice 6.3

3A-NI-08 Explain tradeoffs when selecting and implementing cybersecurity recommendations.
Subconcept: Cybersecurity; Practice 7.2

DATA AND ANALYSIS

3A-DA-09 Translate between different bit representations of real-world phenomena, such as characters, numbers, and images.
Subconcept: Storage; Practice 4.1

3A-DA-10 Evaluate the tradeoffs in how data elements are organized and where data is stored.
Subconcept: Storage; Practice 3.3

3A-DA-11 Create interactive data visualizations using software tools to help others better understand real-world phenomena.
Subconcept: Collection, Visualization & Transformation; Practice 4.4

3A-DA-12 Create computational models that represent the relationships among different elements of data collected from a phenomenon or process.
Subconcept: Inference & Models; Practice 4.4

ALGORITHMS AND PROGRAMMING

3A-AP-13 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
Subconcept: Algorithms; Practice 5.2

3A-AP-14 Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
Subconcept: Variables; Practice 4.1

3A-AP-15 Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.
Subconcept: Control; Practice 5.2

3A-AP-16 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.
Subconcept: Control; Practice 5.2

3A-AP-17 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
Subconcept: Control; Practice 3.2

3A-AP-18 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
Subconcept: Modularity; Practice 5.2

3A-AP-19 Systematically design and develop programs for broad audiences by incorporating feedback from users.
Subconcept: Modularity; Practice 5.1

3A-AP-20 Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.
Subconcept: Program Development; Practice 7.3

3A-AP-21 Evaluate and refine computational artifacts to make them more usable and accessible.
Subconcept: Program Development; Practice 6.3

3A-AP-22 Design and develop computational artifacts working in team roles using collaborative tools.
Subconcept: Program Development; Practice 2.4

3A-AP-23 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.
Subconcept: Program Development; Practice 7.2

IMPACTS OF COMPUTING

3A-IC-24 Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.
Subconcept: Culture; Practice 1.2

3A-IC-25 Test and refine computational artifacts to reduce bias and equity deficits.
Subconcept: Culture; Practice 1.2

3A-IC-26 Demonstrate ways a given algorithm applies to problems across disciplines.
Subconcept: Culture; Practice 3.1

3A-IC-27 Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields.
Subconcept: Social Interactions; Practice 2.4

3A-IC-28 Explain the beneficial and harmful effects that intellectual property laws can have on innovation.
Subconcept: Safety Law & Ethics; Practice 7.3

3A-IC-29 Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.
Subconcept: Safety Law & Ethics; Practice 7.2

3A-IC-30 Evaluate the social and economic implications of privacy in the context of safety, law, or ethics.
Subconcept: Safety Law & Ethics; Practice 7.3

LEVEL 3B: HIGH SCHOOL - SPECIALIZING (GRADES 11-12)

COMPUTING SYSTEMS

3B-CS-01 Categorize the roles of operating system software.
Subconcept: Hardware & Software; Practice 7.2

3B-CS-02 Illustrate ways computing systems implement logic, input, and output through hardware components.
Subconcept: Troubleshooting; Practice 7.2

NETWORKS AND THE INTERNET

3B-NI-03 Describe the issues that impact network functionality (e.g., bandwidth, load, delay, topology).
Subconcept: Network Communication & Organization 7.2

3B-NI-04 Compare ways software developers protect devices and information from unauthorized access.
Subconcept: Cybersecurity; Practice 7.2

DATA AND ANALYSIS

3B-DA-05 Use data analysis tools and techniques to identify patterns in data representing complex systems.
Subconcept: Collection Visualization & Transformation; Practice 4.1

3B-DA-06 Select data collection tools and techniques to generate data sets that support a claim or communicate information.
Subconcept: Collection Visualization & Transformation; Practice 7.2

3B-DA-07 Evaluate the ability of models and simulations to test and support the refinement of hypotheses.
Subconcept: Inference & Models; Practice 4.4

ALGORITHMS AND PROGRAMMING

3B-AP-08 Describe how artificial intelligence drives many software and physical systems.
Subconcept: Algorithms; Practice 7.2

3B-AP-09 Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
Subconcept: Algorithms; Practice 5.3

3B-AP-10 Use and adapt classic algorithms to solve computational problems.
Subconcept: Algorithms; Practice 4.2

3B-AP-11 Evaluate algorithms in terms of their efficiency, correctness, and clarity.
Subconcept: Algorithms; Practice 4.2

3B-AP-12 Compare and contrast fundamental data structures and their uses.
Subconcept: Variables; Practice 4.2

3B-AP-13 Illustrate the flow of execution of a recursive algorithm.
Subconcept: Control; Practice 3.2

3B-AP-14 Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
Subconcept: Modularity; Practice 5.2

3B-AP-15 Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution.
Subconcept: Modularity; Practice 4.1

3B-AP-16 Demonstrate code reuse by creating programming solutions using libraries and APIs.
Subconcept: Modularity; Practice 5.3

3B-AP-17 Plan and develop programs for broad audiences using a software life cycle process.
Subconcept: Program Development; Practice 5.1

3B-AP-18 Explain security issues that might lead to compromised computer programs.
Subconcept: Program Development; Practice 7.2

3B-AP-19 Develop programs for multiple computing platforms.
Subconcept: Program Development; Practice 5.2

3B-AP-20 Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (code documentation) in a group software project.
Subconcept: Program Development; Practice 2.4

3B-AP-21 Develop and use a series of test cases to verify that a program performs according to its design specifications.
Subconcept: Program Development; Practice 6.1

3B-AP-22 Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
Subconcept: Program Development; Practice 5.3

3B-AP-23 Evaluate key qualities of a program through a process such as a code review.
Subconcept: Program Development; Practice 6.3

3B-AP-24 Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems.
Subconcept: Program Development; Practice 7.2

IMPACTS OF COMPUTING

3B-IC-25 Evaluate computational artifacts to maximize their beneficial effects and minimize harmful effects on society.
Subconcept: Culture; Practice 6.1, 1.2

3B-IC-26 Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society.
Subconcept: Culture; Practice 1.2

3B-IC-27 Predict how computational innovations that have revolutionized aspects of our culture might evolve.
Subconcept: Culture; Practice 7.2

3B-IC-28 Debate laws and regulations that impact the development and use of software.
Subconcept: Safety Law & Ethics; Practice 3.3, 7.3

REFERENCES

Computer Science Teachers Association. CSTA Standards. Retrieved from https://www.csteachers.org/page/standards

Digital Promise. Computational Thinking for a Computational World. Retrieved from http://digitalpromise.org/wp-content/uploads/2018/05/dp-comp-thinking-v1r6.pdf

Google & Gallop (2015). Images of Computer Science: Perceptions among Students, Parents, and Educators in the U.S. Retrieved from https://services.google.com/fh/files/misc/images-of-computer-science-report.pdf

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

Michigan Department of Education. Michigan's Top 10 in 10 Years. Retrieved from https://www.michigan.gov/documents/mde/10in10_FINAL_507948_7.pdf

Michigan Department of Education. MI Roadmap. Retrieved from http://www.techplan.org/mi-roadmap/

Michigan Department of Education. Michigan Integrated Technology Competencies for Students (MITECS). Retrieved from http://www.techplan.org/mitecs/

PRN Newswire. (October 10, 2015). Horizon Media Study Reveals Americans Prioritize STEM Subjects over the Arts; Science is "Cool," Coding is New Literacy. Retrieved from https://www.prnewswire.com/news-releases/horizon-media-study-reveals-americans-prioritize-stem-subjects-over-the-arts-science-is-cool-coding-is-new-literacy-300154137.html

PROGRESSION OF MICHIGAN COMPUTER SCIENCE STANDARDS

CONCEPT: COMPUTING SYSTEMS

Subconcept	Level 1A: Lower Elementary (Grades K-2) By the end of Grade 2, students will be able to...	Level 1B: Upper Elementary (Grades 3-5) By the end of Grade 5, students will be able to...
Devices	1A-CS-01 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. (P1.1)	1B-CS-01 Describe how internal and external parts of computing devices function to form a system. (P7.2)
Hardware & Software	1A-CS-02 Use appropriate terminology in identifying and describing the function of common physical components of computing systems (hardware). (P7.2)	1B-CS-02 Model how computer hardware and software work together as a system to accomplish tasks. (P4.4)
Troubleshooting	1A-CS-03 Describe basic hardware and software problems using accurate terminology. (P6.2, P7.2)	1B-CS-03 Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies. (P6.2)

Subconcept	Level 2: Middle School (Grades 6-8) By the end of Grade 8, students will be able to...	Level 3a: High School (Grades 9-10) By the end of Grade 10, students will be able to...
Devices	2-CS-01 Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices. (P3.3)	3A-CS-01 Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects. (P4.1)
Hardware & Software	2-CS-02 Design projects that combine hardware and software components to collect and exchange data. (P5.1)	3A-CS-02 Compare levels of abstraction and interactions between application software, system software, and hardware layers. (P4.1)
Troubleshooting	2-CS-03 Systematically identify and fix problems with computing devices and their components. (P6.2)	3A-CS-03 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors. (P6.2)

CONCEPT: NETWORKS & THE INTERNET

Subconcept	Level 1A: Lower Elementary (Grades K-2) By the end of Grade 2, students will be able to...	Level 1B: Upper Elementary (Grades 3-5) By the end of Grade 5, students will be able to...
Network Communication & Organization	N/A	1B-NI-04 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. (P4.4)
Cybersecurity	1A-NI-04 Explain what passwords are and why we use them, and use strong passwords to protect devices and information from unauthorized access. (P7.3)	1B-NI-05 Discuss real-world cybersecurity problems and how personal information can be protected. (P3.1)

Subconcept	Level 2: Middle School (Grades 6-8) By the end of Grade 8, students will be able to...	Level 3a: High School (Grades 9-10) By the end of Grade 10, students will be able to...
Network Communication & Organization	2-NI-04 Model the role of protocols in transmitting data across networks and the Internet. (P4.4)	3A-NI-04 Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing. (P4.1)
Cybersecurity	2-NI-05 Explain how physical and digital security measures protect electronic information. (P7.2) 2-NI-06 Apply multiple methods of encryption to model the secure transmission of information. (P4.4)	3A-NI-05 Give examples to illustrate how sensitive data can be affected by malware and other attacks. (P7.2) 3A-NI-06 Recommend security measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts. (P3.3) 3A-NI-07 Compare various security measures, considering tradeoffs between the usability and security of a computing system. (P6.3) 3A-NI-08 Explain tradeoffs when selecting and implementing cybersecurity recommendations. (P7.2)

CONCEPT: DATA & ANALYSIS

Subconcept	Level 1A: Lower Elementary (Grades K-2) By the end of Grade 2, students will be able to...	Level 1B: Upper Elementary (Grades 3-5) By the end of Grade 5, students will be able to...
Storage	1A-DA-05 Store, copy, search, retrieve, modify, and delete information using a computing device and define the information stored as data. (P4.2)	Continuation of standard 1A-DA-05
Collection, Visualization, & Transformation	1A-DA-06 Collect and present the same data in various visual formats. (P7.1, P4.4)	1B-DA-06 Organize and present collected data visually to highlight relationships and support a claim. (P7.1)
Inference & Models	1A-DA-07 Identify and describe patterns in data visualizations, such as charts or graphs, to make predictions. (P4.1)	1B-DA-07 Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea. (P7.1)

Subconcept	Level 2: Middle School (Grades 6-8) By the end of Grade 8, students will be able to...	Level 3a: High School (Grades 9-10) By the end of Grade 10, students will be able to...
Storage	2-DA-07 Represent data using multiple encoding schemes. (P4.0)	3A-DA-09 Translate between different bit representations of real-world phenomena, such as characters, numbers, and images. (P4.1) 3A-DA-10 Evaluate the tradeoffs in how data elements are organized and where data is stored. (P3.3)
Collection, Visualization, & Transformation	2-DA-08 Collect data using computational tools and transform the data to make it more useful and reliable. (P6.3)	3A-DA-11 Create interactive data visualizations using software tools to help others better understand realworld phenomena. (P4.4)
Inference & Models	2-DA-09 Refine computational models based on the data they have generated. (P5.3, P4.4)	3A-DA-12 Create computational models that represent the relationships among different elements of data collected from a phenomenon or process. (P4.4)

CONCEPT: ALGORITHMS & PROGRAMMING

Subconcept	Level 1A: Lower Elementary (Grades K-2) By the end of Grade 2, students will be able to...	Level 1B: Upper Elementary (Grades 3-5) By the end of Grade 5, students will be able to...
Algorithms	1A-AP-08 Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks. (P4.4)	1B-AP-08 Compare and refine multiple algorithms forthe same task and determine which is the most appropriate. (P6.3, P3.3)
Variables	1A-AP-09 Model the way programs store and manipulate data by using numbers or other symbols to represent information. (P4.4)	1B-AP-09 Create programs that use variables to store and modify data. (P5.2)
Control	1A-AP-10 Develop programs with sequences and simple loops, to express ideas or address a problem. (P5.2)	1B-AP-10 Create programs that include sequences, events, loops, and conditionals. (P5.2)
Modularity	1A-AP-11 Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions. (P3.2)	1B-AP-11 Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process. (P3.2) 1B-AP-12 Modify, remix, or incorporate portions of an existing program into one's own work, to develop something new or add more advanced features. (P5.3)

Subconcept	Level 1A: Lower Elementary (Grades K-2) By the end of Grade 2, students will be able to...	Level 1B: Upper Elementary (Grades 3-5) By the end of Grade 5, students will be able to...
Program Development	1A-AP-12 Develop plans that describe a program's sequence of events, goals, and expected outcomes. (P5.1, P7.2) 1A-AP-13 Give attribution when using the ideas and creations of others while developing programs. (P7.3) 1A-AP-14 Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops. (P6.2) 1A-AP-15 Using correct terminology, describe steps taken and choices made during the iterative process of program development. (P7.2)	1B-AP-13 Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences. (P1.1, P5.1) 1B-AP-14 Observe intellectual property rights and give appropriate attribution when creating or remixing programs. (P7.3) 1B-AP-15 Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended. (P6.1, P6.2) 1B-AP-16 Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development. (P2.2) 1B-AP-17 Describe choices made during program development using code comments, presentations, and demonstrations. (P7.2)

Subconcept	Level 2: Middle School (Grades 6-8) By the end of Grade 8, students will be able to...	Level 3a: High School (Grades 9-10) By the end of Grade 10, students will be able to...
Algorithms	2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms. (P4.4, P4.1)	3A-AP-13 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests. (P5.2)
Variables	2-AP-11 Create clearly named variables that represent different data types and perform operations on their values. (P5.1, P5.2)	3A-AP-14 Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables. (P4.1)
Control	2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals. (P5.1, P5.2)	3A-AP-15 Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made. (P5.2) 3A-AP-16 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions. (P5.2)
Modularity	2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. (P3.2) 2-AP-14 Create procedures with parameters to organize code and make it easier to reuse. (P4.1, P4. 3)	3A-AP-17 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects. (P3.2) 3A-AP-18 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs. (P5.2)
Program Development	2-AP-15 Seek and incorporate feedback from team members and users to refine a solution that meets user needs. (P2.3, P1.1) 2-AP-16 Incorporate existing code, media, and libraries into original programs, and give attribution. (P4.2, P5.2, P7.3) 2-AP-17 Systematically test and refine programs using a range of test cases. (P6.1) 2-AP-18 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts. (P2.2) 2-AP-19 Document programs in order to make them easier to follow, test, and debug. (P7.2)	3A-AP-19 Systematically design and develop programs for broad audiences by incorporating feedback from users. (P5.1) 3A-AP-20 Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries. (P7.3) 3A-AP-21 Evaluate and refine computational artifacts to make them more usable and accessible. (P6.3) 3A-AP-22 Design and develop computational artifacts working in team roles using collaborative tools. (P2.4) 3A-AP-23 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs. (P7.2)

CONCEPT: IMPACTS OF COMPUTING

Subconcept	Level 1A: Lower Elementary (Grades K-2) By the end of Grade 2, students will be able to...	Level 1B: Upper Elementary (Grades 3-5) By the end of Grade 5, students will be able to...
Culture	1A-IC-16 Compare how people live and work before and after the implementation or adoption of new computing technology. (P7.0)	1B-IC-18 Discuss computing technologies that have changed the world, and express how those technologies influence, and are influenced by, cultural practices. (P7.1) 1B-IC-19 Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users. (P1.2)
Social Interactions	1A-IC-17 Work respectfully and responsibly with others online. (P2.1)	1B-IC-20 Seek diverse perspectives for the purpose of improving computational artifacts. (P1.1)
Safety, Law, & Ethics	1A-IC-18 Keep login information private, and log off of devices appropriately. (P7.3)	1B-IC-21 Use public domain or creative commons media, and refrain from copying or using material created by others without permission. (P7.3)

Subconcept	Level 2: Middle School (Grades 6-8) By the end of Grade 8, students will be able to...	Level 3a: High School (Grades 9-10) By the end of Grade 10, students will be able to...
Culture	2-IC-20 Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options. (P7.2) 2-IC-21 Discuss issues of bias and accessibility in the design of existing technologies. (P1.2)	3A-IC-24 Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices. (P1.2) 3A-IC-25 Test and refine computational artifacts to reduce bias and equity deficits. (P1.2) 3A-IC-26 Demonstrate ways a given algorithm applies to problems across disciplines. (P3.1)
Social Interactions	2-IC-22 Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact. (P2.4, P5.2)	3A-IC-27 Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields. (P2.4)
Safety, Law, & Ethics	2-IC-23 Describe tradeoffs between allowing information to be public and keeping information private and secure. (P7.2)	3A-IC-28 Explain the beneficial and harmful effects that intellectual property laws can have on innovation. (P7.3) 3A-IC-29 Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users. (P7.2) 3A-IC-30 Evaluate the social and economic implications of privacy in the context of safety, law, or ethics. (P7.3)

Practices: P1. Fostering an Inclusive Computing Culture; P2. Collaborating Around Computing; P3. Recognizing and Defining Computational Problems; P4. Developing and Using Abstractions; P5. Creating Computational Artifacts; P6. Testing and Refining Computational Artifacts; P7. Communicating About Computing

GLOSSARY

The glossary includes definitions of terms used in the statements in the framework. These terms are defined for readers of the framework and are not necessarily intended to be the definitions or terms that are seen by students.

TABLE C.1: GLOSSARY TERMS

Term	Definition
Abstraction	(process): The process of reducing complexity by focusing on the main idea. By hiding details irrelevant to the question at hand and bringing together related and useful details, abstraction reduces complexity and allows one to focus on the problem. (product): A new representation of a thing, a system, or a problem that helpfully reframes a problem by hiding details irrelevant to the question at hand. [MDESE, 2016]
Accessibility	The design of products, devices, services, or environments for people who experience disabilities. Accessibility standards that are generally accepted by professional groups include the Web Content Accessibility Guidelines (WCAG) 2.0 and Accessible Rich Internet Applica- tions (ARIA) standards. [Wikipedia]
Algorithm	A step-by-step process to complete a task.
Analog	The defining characteristic of data that is represented in a continuous, physical way. Whereas digital data is a set of individual symbols, analog data is stored in physical media, such as the surface grooves on a vinyl record, the magnetic tape of a VCR cassette, or other nondigital media. [Techopedia]
App	A type of application software designed to run on a mobile device, such as a smartphone or tablet computer. Also known as a mobile application. [Techopedia]
Artifact	Anything created by a human. See computational artifact for the definition used in computer science.
Audience	Expected end users of a computational artifact or system.
Accessibility	The design of products, devices, services, or environments for people who experience disabilities. Accessibility standards that are generally accepted by professional groups include the Web Content Accessibility Guidelines (WCAG) 2.0 and Accessible Rich Internet Applica- tions (ARIA) standards. [Wikipedia]
Authentication	The verification of the identity of a person or process. [FOLDOC]
Automate; Automation	automate: To link disparate systems and software so that they become self-acting or self-regu- lating. [Ross, 2016] automation: The process of automating.
Bit Representations	i.e. characters, numbers, and images (3A-DA-09)
Boolean	A type of data or expression with two possible values: true and false. [FOLDOC]
Bug	An error in a software program. It may cause a program to unexpectedly quit or behave in an unintended manner. [Tech Terms] The process of finding and correcting errors (bugs) is called debugging. [Wikipedia]
Code	Any set of instructions expressed in a programming language. [MDESE, 2016]
Comment	A programmer-readable annotation in the code of a computer program added to make the code easier to understand. Comments are generally ignored by machines. [Wikipedia]
Complexity	The minimum amount of resources, such as memory, time, or messages, needed to solve a problem or execute an algorithm. [NIST/DADS]
Component	An element of a larger group. Usually, a component provides a particular service or group of related services. [Tech Terms, TechTarget]

Term	Definition
Compound Conditionals	Compound conditionals combine two or more conditions in a logical relationship (e.g., using AND, OR, and NOT), and nesting conditions within one another allows the result of one conditional to lead to another.
Computational	Relating to computers or computing methods.
Computational Artifact	Anything created by a human using a computational thinking process and a computing device. A computational artifact can be, but is not limited to, a program, image, audio, video, presentation, or web page file. [College Board, 2016]
Computational Thinking	The human ability to formulate problems so that their solutions can be represented as computational steps or algorithms to be executed by a computer. [Lee, 2016]
Computer	A machine or device that performs processes, calculations, and operations based on instruc- tions provided by a software or hardware program. [Techopedia]
Computer Science	The study of computers and algorithmic processes, including their principles, their hardware and software designs, their implementation, and their impact on society. [ACM, 2006]
Computing	Any goal-oriented activity requiring, benefiting from, or creating algorithmic processes. [MDESE, 2016]
Computing Device	A physical device that uses hardware and software to receive, process, and output information. Computers, mobile phones, and computer chips inside appliances are all examples of computing devices.
Computing System	A collection of one or more computers or computing devices, together with their hardware and software, integrated for the purpose of accomplishing shared tasks. Although a computing system can be limited to a single computer or computing device, it more commonly refers to a collection of multiple connected computers, computing devices, and hardware.
Conditional	A feature of a programming language that performs different computations or actions depend- ing on whether a programmer-specified Boolean condition evaluates to true or false. [MDESE, 2016] (A conditional could refer to a conditional statement, conditional expression, or conditional construct.)
Configuration	(process): Defining the options that are provided when installing or modifying hardware and software or the process of creating the configuration (product). [TechTarget] (product): The specific hardware and software details that tell exactly what the system is made up of, especially in terms of devices attached, capacity, or capability. [TechTarget]
Connection	A physical or wireless attachment between multiple computing systems, computers, or computing devices.
Connectivity	A program's or device's ability to link with other programs and devices. [Webopedia]
Control; Control Structure	control: (in general) The power to direct the course of actions. (in programming) The use of elements of programming code to direct which actions take place and the order in which they take place. control structure: A programming (code) structure that implements control. Conditionals and loops are examples of control structures.
Culture; Cultural Practices	culture: A human institution manifested in the learned behavior of people, including their specific belief systems, language(s), social relations, technologies, institutions, organizations, and systems for using and developing resources. [NCSS, 2013] cultural practices: The displays and behaviors of a culture.
Cybersecurity	The protection against access to, or alteration of, computing resources through the use of technology, processes, and training. [TechTarget]

Term	Definition
Data	Information that is collected and used for reference or analysis. Data can be digital or nondigi- tal and can be in many forms, including numbers, text, show of hands, images, sounds, or video. [CAS, 2013; Tech Terms]
Data Structure	A particular way to store and organize data within a computer program to suit a specific purpose so that it can be accessed and worked with in appropriate ways. [TechTarget]
Data Type	A classification of data that is distinguished by its attributes and the types of operations that can be performed on it. Some common data types are integer, string, Boolean (true or false), and floating-point.
Debugging	The process of finding and correcting errors (bugs) in programs. [MDESE, 2016]
Decompose; Decomposition	decompose: To break down into components. decomposition: Breaking down a problem or system into components. [MDESE, 2016]
Device	A unit of physical hardware that provides one or more computing functions within a computing system. It can provide input to the computer, accept output, or both. [Techopedia]
Digital	A characteristic of electronic technology that uses discrete values, generally 0 and 1, to generate, store, and process data. [Techopedia]
Digital Citizenship	The norms of appropriate, responsible behavior with regard to the use of technology. [MDESE, 2016]
Efficiency	A measure of the amount of resources an algorithm uses to find an answer. It is usually expressed in terms of the theoretical computations, the memory used, the number of messages passed, the number of disk accesses, etc. [NIST/DADS]
Encapsulation	The technique of combining data and the procedures that act on it to create a type. [FOLDOC]
Encoding Schemes	Representing the same data in multiple ways, i.e., representing the same color using binary, RGB values, or hex codes.
Encryption	The conversion of electronic data into another form, called ciphertext, which cannot be easily understood by anyone except authorized parties. [TechTarget]
End User (or User)	A person for whom a hardware or software product is designed (as distinguished from the developers). [TechTarget]
Event	Any identifiable occurrence that has significance for system hardware or software. User-gener- ated events include keystrokes and mouse clicks; system-generated events include program loading and errors. [TechTarget]
Event Handler	A procedure that specifies what should happen when a specific event occurs.
Execute; Execution	execute: To carry out (or "run") an instruction or set of instructions (program, app, etc.). execution: The process of executing an instruction or set of instructions. [FOLDOC]
Hardware	The physical components that make up a computing system, computer, or computing device. [MDESE, 2016]
Hierarchy	An organizational structure in which items are ranked according to levels of importance. [TechTarget]
Human–Computer Interaction (HCI)	The study of how people interact with computers and to what extent computing systems are or are not developed for successful interaction with human beings. [TechTarget]
Identifier	The user-defined, unique name of a program element (such as a variable or procedure) in code. An identifier name should indicate the meaning and usage of the element being named. [Techopedia]
Implementation	The process of expressing the design of a solution in a programming language (code) that can be made to run on a computing device.
Inference	A conclusion reached on the basis of evidence and reasoning. [Oxford]

Term	Definition
Input	The signals or instructions sent to a computer. [Techopedia]
Integrity	The overall completeness, accuracy, and consistency of data. [Techopedia]
Intellectual Property Laws	Laws governing many aspects of computing, such as privacy, data, property, information, and identity. These laws can have beneficial and harmful effects, such as expediting or delaying advancements in computing and protecting or infringing upon people's rights.
Internet	The global collection of computer networks and their connections, all using shared protocols to communicate. [CAS, 2013]
Iterative	Involving the repeating of a process with the aim of approaching a desired goal, target, or result. [MDESE, 2016]
Libraries and Application Program Interfaces (API)	a collection of commands made available to a programmer; a collection of commands / functions, typically with a shared purpose, specific to a programming language
Loop	A programming structure that repeats a sequence of instructions as long as a specific condition is true. [Tech Terms]
Memory	Temporary storage used by computing devices. [MDESE, 2016]
Model	A representation of some part of a problem or a system. [MDESE, 2016] Note: This definition differs from that used in science.
Modularity	The characteristic of a software/web application that has been divided (decomposed) into smaller modules. An application might have several procedures that are called from inside its main procedure. Existing procedures could be reused by recombining them in a new applica- tion. [Techopedia]
Module	A software component or part of a program that contains one or more procedures. One or more independently developed modules make up a program. [Techopedia]
Network	A group of computing devices (personal computers, phones, servers, switches, routers, etc.) connected by cables or wireless media for the exchange of information and resources.
Operation	An action, resulting from a single instruction, that changes the state of data. [Free Dictionary]
Packet	The unit of data sent over a network. [Tech Terms]
Parameter	A special kind of variable used in a procedure to refer to one of the pieces of data received as input by the procedure. [MDESE, 2016]
Physical and Digital Security	Methods to protect data from vulnerability. Physical security examples include keeping passwords hidden, locking doors, making backup copies on external storage devices, and erasing a storage device before it is reused. Examples of digital security measures include secure router admin passwords, firewalls that limit access to private networks, and the use of a protocol such as HTTPS to ensure secure data transmission.
Piracy	The illegal copying, distribution, or use of software. [TechTarget]
Procedure	An independent code module that fulfills some concrete task and is referenced within a larger body of program code. The fundamental role of a procedure is to offer a single point of reference for some small goal or task that the developer or programmer can trigger by invoking the procedure itself. [Techopedia] In this framework, procedure is used as a general term that may refer to an actual procedure or a method, function, or module of any other name by which modules are known in other programming languages.
Process	A series of actions or steps taken to achieve a particular outcome. [Oxford]

Term	Definition
Program; Programming	program (n): A set of instructions that the computer executes to achieve a particular objective. [MDESE, 2016] program (v): To produce a program by programming. programming: The craft of analyzing problems and designing, writing, testing, and maintain- ing programs to solve them. [MDESE, 2016]
Protocol	The special set of rules used by endpoints in a telecommunication connection when they communicate. Protocols specify interactions between the communicating entities. [TechTarget]
Prototype	An early approximation of a final product or information system, often built for demonstration purposes. [TechTarget, Techopedia]
Pseudocode	A plain English notation that resembles the structural conventions of programming languages, typically used to reason about programs and algorithms
Redundancy	A system design in which a component is duplicated, so if it fails, there will be a backup. [TechTarget]
Reliability	An attribute of any system that consistently produces the same results, preferably meeting or exceeding its requirements. [FOLDOC]
Remix	The process of creating something new from something old. Originally a process that involved music, remixing involves creating a new version of a program by recombining and modifying parts of existing programs, and often adding new pieces, to form new solutions. [Kafai & Burke, 2014]
Router	A device or software that determines the path that data packets travel from source to destination. [TechTarget]
Scalability	The capability of a network to handle a growing amount of work or its potential to be enlarged to accommodate that growth. [Wikipedia]
Security	See the definition for cybersecurity.
Simulate; Simulation	simulate: To imitate the operation of a real-world process or system. simulation: Imitation of the operation of a real-world process or system. [MDESE, 2016]
Software	Programs that run on a computing system, computer, or other computing device.
Software Life Cycle Process	A project's development from inception to delivery, to maintenance, to updating, and delivery again… to its eventual obsolescence or retirement
Storage	(place) A place, usually a device, into which data can be entered, in which the data can be held, and from which the data can be retrieved at a later time. [FOLDOC] (process) A process through which digital data is saved within a data storage device by means of computing technology. Storage is a mechanism that enables a computer to retain data, either temporarily or permanently. [Techopedia]
String	A sequence of letters, numbers, and/or other symbols. A string might represent, for example, a name, address, or song title. Some functions commonly associated with strings are length, concatenation, and substring. [TechTarget]
Structure	A general term used in the framework to discuss the concept of encapsulation without specifying a particular programming methodology.
Switch	A high-speed device that receives incoming data packets and redirects them to their destina- tion on a local area network (LAN). [Techopedia]
System	A collection of elements or components that work together for a common purpose. [TechTar- get] See also the definition for computing system.
Test Case	A set of conditions or variables under which a tester will determine whether the system being tested satisfies requirements or works correctly. [STF]

Term	Definition
Topology	The physical and logical configuration of a network; the arrangement of a network, including its nodes and connecting links. A logical topology is the way devices appear connected to the user. A physical topology is the way they are actually interconnected with wires and cables. [PCMag]
Troubleshooting	A systematic approach to problem solving that is often used to find and resolve a problem, error, or fault within software or a computing system. [Techopedia, TechTarget]
User	See the definition for end user.
Variable	A symbolic name that is used to keep track of a value that can change while a program is running. Variables are not just used for numbers; they can also hold text, including whole sentences (strings) or logical values (true or false). A variable has a data type and is associated with a data storage location; its value is normally changed during the course of program execution. [CAS, 2013; Techopedia] Note: This definition differs from that used in math.

GLOSSARY REFERENCES

TABLE C.2: GLOSSARY REFERENCES

Publication	Citation
ACM, 2006	A Model Curriculum for K–12 Computer Science Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). A model curriculum for K–12 computer science: Report of the ACM K–12 task force curriculum commit- tee (2nd ed.). New York, NY: Association for Computing Machinery.
CAS, 2013	Computing At School's Computing in the National Curriculum: A Guide for Primary Teachers Computing At School. (2013). Computing in the national curriculum: A guide for primary teachers. Belford, UK: Newnorth Print. Retrieved from http://www.computingatschool.org.uk/ data/uploads/CASPrimaryComputing.pdf
College Board, 2016	College Board Advanced Placement® Computer Science Principles College Board. (2016). AP Computer Science Principles course and exam description. New York, NY: College Board. Retrieved from https://secure-media.collegeboard.org/digitalServices/ pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf
FOLDOC	Free On-Line Dictionary of Computing Free on-line dictionary of computing. (n.d.). Retrieved from http://foldoc.org
Free Dictionary	The Free Dictionary The free dictionary. (n.d.). Retrieved from http://www.thefreedictionary.com
Kafai & Burke, 2014	Connected Code: Why Children Need to Learn Programming Kafai, Y., & Burke, Q. (2014). Connected code: Why children need to learn programming. Cambridge, MA: MIT Press.
Lee, 2016	Reclaiming the Roots of CT Lee, I. (2016). Reclaiming the roots of CT. CSTA Voice: The Voice of K–12 Computer Science Education and Its Educators, 12(1), 3–4. Retrieved from http://www.csteachers.org/resource/ resmgr/Voice/csta_voice_03_2016.pdf
MDESE, 2016	Massachusetts Digital Literacy and Computer Science (DL&CS) Standards Massachusetts Department of Elementary and Secondary Education. (2016, June). 2016 Massachusetts digital literacy and computer science (DLCS) curriculum framework. Malden, MA: Author. Retrieved from http://www.doe.mass.edu/frameworks/dlcs.pdf
NCSS, 2013	College, Career & Civic Life (C3) Framework for Social Studies State Standards National Council for the Social Studies. (2013). The college, career, and civic life (C3) framework for social studies state standards: Guidance for enhancing the rigor of K–12 civics, economics, geography, and history. Silver Spring, MD: Author. Retrieved from http://www.socialstudies.org/ system/files/c3/C3-Framework-for-Social-Studies.pdf
NIST/DADS	National Institute of Science and Technology Dictionary of Algorithms and Data Structures Pieterse, V., & Black, P. E. (Eds.). (n.d). Dictionary of algorithms and data structures. Retrieved from https://xlinux.nist.gov/dads//
Oxford	Oxford Dictionaries Oxford dictionaries. (n.d.). Retrieved from http://www.oxforddictionaries.com/us
PCmag	PCmag.com Encyclopedia PCmag.com encyclopedia. (n.d.). Retrieved from http://www.pcmag.com/encyclopedia/term/46301/logical-vs-physical-topology

Publication	Citation
Ross, 2016	What Is Automation Ross, B. (2016, May 10). What is automation and how can it improve customer service? Information Age. Retrieved from http://www.information-age.com/industry/software/123461408/what-automation-and-how-can-it-improve-customer-service
STF	Software Testing Fundamentals Software testing fundamentals. (n.d). Retrieved from http://softwaretestingfundamentals.com
Tech Terms	Tech Terms Tech terms computer dictionary. (n.d.). Retrieved from http://www.techterms.com
Techopedia	Techopedia Techopedia technology dictionary. (n.d.). Retrieved from https://www.techopedia.com/dictionary
TechTarget	TechTarget Network TechTarget network. (n.d.). Retrieved from http://www.techtarget.com/network
Webopedia	Webopedia Webopedia. (n.d.). Retrieved from http://www.webopedia.com

Michigan State Board of Education

Casandra E. Ulbrich
President

Pamela Pugh
Vice President

Michelle Fecteau
Secretary

Tom McMillin
Treasurer

Judith Pritchett

Lupe Ramos-Montigny

Nikki Snyder

Tiffany Tilley
NASBE Delegate

The Honorable Gretchen Whitmer
Governor
Ex Officio

Ms. Sheila A. Alles
Chairman
Ex Officio
Interim State Superintendent

MDE Staff

Venessa A. Keesler, Ph.D.
Deputy Superintendent
Educator, Student, and School Supports

David A. Judd, Director
Office of Systems, Evaluation, and Technology

Michigan Department of Education
Office of Systems, Evaluation,
and Technology
p: (517) 241-6966
website: www.michigan.gov/mde

MICHIGAN Department of Education

COMPUTER SCIENCE STANDARDS STAKEHOLDER COMMITTEE

Eric Bently, Perry Public Schools

Ryan Cayce, Michigan Association of Secondary School Principals (MASSP)

Melanie Cochrill, New Haven High School

Steve Dickie, Michigan Association of Computer Users in Learning (MACUL)

Lori Flippin, Great Lakes Bay Regional Alliance

Ted Gardella, College Board

Carlos Garcia, Jackson College

Tony Garcia, Michigan Film & Digital Media Office

Selam Ghirmai, Michigan Film & Digital Media Office

Paul Groll, Michigan Department of Technology Management and Budget

Colleen Gubow, Early Childhood Representative, Hazel Park Schools

Ruth Ann Hodges, Michigan Department of Education (MDE), Office of Educational Supports

Shanika Hope, Amazon Web Services

Joanne Hopper, Facilitator

Dave Judd, Michigan Department of Education (MDE), Office of Systems, Evaluation & Technology

Vinos Kassab, Oakland Schools, Michigan Computer Science Teachers' Association

Heidi Kattula, East Grand Rapids Schools, Michigan Computer Science Teachers' Association

Tom Knight, Michigan Department of Education (MDE), Career and Technical Education

Dave Krebs, General Education Leadership Network

Ann-Marie Mapes, Michigan Department of Education (MDE), Office of Systems, Evaluation & Technology

Lynn McNamara, Michigan Film & Digital Media Office

Gabriella Meyers, Apple

Dave Parmele, Amazon Web Services

Josh Pudaloff, Michigan Computer Science Teachers' Association, Troy School District

Michelle Ribant, Michigan Department of Education (MDE), Office of Systems, Evaluation & Technology

Sean Roberts, Code.org

Kevin Santer, Michigan Virtual

Tyler Sawher, Office of Governor

Snyder

Aaron Schippert, Michigan Educational Technology Leaders (METL), Saginaw ISD

Megan Schrauben, MI STEM Network, Department of Technology Management and Budget

Patrick Schultz, Bay-Arenac ISD, Michigan Initiative for Cybersecurity Education (MICE)

Pietro Semifero, Michigan Department of Education (MDE), Office of Educational Assessment and Accountability

Matt Shastal, Michigan Association of School Administrators (MASA)

Mark Smith, Michigan Association of Computer Users in Learning (MACUL)

Andew Spiece, Microsoft Philanthropies TEALS

Amanda Stoel, Michigan Department of Education (MDE), Office of Systems, Evaluation & Technology

Richard Straughen, Utica Community Schools

Kathy Surd, West Central STEM Region, Code.org Regional Manager

Zachary Sweet, Detroit Public Schools Community District

Joe Trommater, Clare/Gladwin Regional Educational Service District

Mike Tucker, Square One Education Network

Aman Yadav, Michigan State University

Alexandra Orellano-Vlachakis, Detroit Public Schools Community District

Melinda Waffle, Regional Educational Media Center (REMC), Calhoun Intermediate School District

Charley Williams, Code.org

ADVISORS

Allyson Knox, Microsoft

Daniel Moix, Director, Coding Arkansas' Future at Arkansas School for Math, Sciences and the Arts

Jason Molesky, Apple

Dr. David Richards, President, Core2Edge

Dave Seitz, Apple

Chris Stephenson, Head of Computer Science Education Strategy, Google

Catherine Woolman, Executive Director of Instruction, Port Huron Area School District