6-8 Crosswalk (CSTA Reviewer Grant Smith)

6-8 Final	Level 2	6-8 Interim	Level 2
2-CS-01	Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.	2-A-7-4	Interpret the flow of execution of algorithms and predict their outcomes. [Clarification: Algorithms can be expressed using natural language, flow and control diagrams, comments within code, and pseudocode.]
2-CS-02	Design projects that combine hardware and software components to collect and exchange data.	2-C-7-11	Justify the hardware and software chosen to accomplish a task (e.g., comparison of the features of a tablet vs. desktop, selecting which sensors and platform to use in building a robot or developing a mobile app).
2-CS-03	Systematically identify and fix problems with computing devices and their components.	2-C-6-13	Use a systematic process to identify the source of a problem within individual and connected devices (e.g., follow a troubleshooting flow diagram, make changes to software to see if hardware will work, restart device, check connections, swap in working components).
2-NI-04	Model the role of protocols in transmitting data across networks and the Internet.	2-N-4-25	Simulate how information is transmitted as packets through multiple devices over the Internet and Networks.
2-NI-05	Explain how physical and digital security measures protect electronic information.	2-I-1-22	Describe ethical issues that relate to computing devices and networks (e.g., equity of access, security and plagiarism), hacking, intellectual property, copyright, Creative Commons licensing.
2-NI-06	Apply multiple methods of encryption to model the secure transmission of information.		
2-DA-07	Represent data using multiple encoding schemes.	2-D-4-17	Represent data using different encoding schemes (e.g., binary, Unicode, Morse code, shorthand, student-created codes).
2-DA-08	Collect data using computational tools and transform the data to make it more useful and reliable.	2-D-7-15	Explain the processes used to collect, transform, and analyze data to solve a problem using computational tools (e.g., use an app or spreadsheet form to collect data, decide which data to use or ignore, and choose a visualization method.).
2-DA-09	Refine computational models based on the data they have generated.		New

6-8 Final	Level 2	6-8 Interim	Level 2
2-AP-10	Use flowcharts and/or pseudocode to address complex problems as algorithms.	2-A-6-10	Use an iterative design process (e.g., define the problem, generate ideas, build, test, and improve solutions) to solve problems, both independently and collaboratively.
2-AP-11	Create clearly named variables that represent different data types and perform operations on their values.	2-A-5-7	Create variables that represent different types of data and manipulate their values.
2-AP-12	Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.	2-A-5-6	Develop programs, both independently and collaboratively, that include sequences with nested loops and multiple branches. [Clarification: At this level, students may use block-based and/or text-based programming languages.]
2-AP-13	Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.	2-A-3-9	Decompose a problem into parts and create solutions for each part.
2-AP-14	Create procedures with parameters to organize code and make it easier to reuse.	2-A-4-8	Define and use procedures that hide the complexity of a task and can be reused to solve similar tasks. [Clarification: Students use and modify, but do not necessarily create, procedures with parameters.]
2-AP-15	Seek and incorporate feedback from team members and users to refine a solution that meets user needs.	2-A-2-1	Solicit and integrate peer feedback as appropriate to develop or refine a program.
2-AP-16	Incorporate existing code, media, and libraries into original programs, and give attribution.	2-A-7-3	Provide proper attribution when code is borrowed or built upon.
2-AP-17	Systematically test and refine programs using a range of test cases.	2-D-5-16	Revise computational models to more accurately reflect real-world systems (e.g., ecosystems, epidemics, spread of ideas).
2-AP-18	Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.	2-A-5-5	Design, develop, and present computational artifacts such as mobile applications that address social problems both independently and collaboratively.
2-AP-19	Document programs in order to make them easier to follow, test, and debug.		New
2-IC-20	Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.	2-C-4-12	Analyze the relationship between a device's computational components and its capabilities. [Clarification: Computing Systems include not only computers, but also cars, microwaves, smartphones, traffic lights, and flash drives.]

6-8 Final	Level 2	6-8 Interim	Level 2
2-IC-21	Discuss issues of bias and accessibility in the design of existing technologies.	2-I-7-18	Summarize negative and positive impacts of using data and information to categorize people, predict behavior, and make recommendations based on those predictions (e.g., customizing search results or targeted advertising, based on previous browsing history, can save search time and limit options at the same time).
		2-I-6-23	Redesign a computational artifact to remove barriers to universal access (e.g., using captions on images, high contrast colors, and/or larger font sizes).
2-IC-22	Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.	2-I-7-19	Explain how computer science fosters innovation and enhances nearly all careers and disciplines.
2-IC-23	Describe tradeoffs between allowing information to be public and keeping information private and secure.	2-N-7-24	Summarize security risks associated with weak passwords, lack of encryption, insecure transactions, and persistence of data.
	Removed	2-A-7-2	Compare different algorithms that may be used to solve the same problem, but one might be faster than the other. (e.g., different algorithms solve the same problem, but one might be faster than the other). [Clarification: Students are not expected to quantify these differences.]
	Removed	2-D-7-14	Describe how different formats of stored data represent tradeoffs between quality and size. [Clarification: compare examples of music, text and/or image formats.]
	Removed	2-I-1-20	Provide examples of how computational artifacts and devices impact health and wellbeing, both positively and negatively.
	Removed	2-I-1-21	Describe ways in which the Internet impacts global communication and collaborating.

REVERSED TABLE

6-8 Interim	Level 2	6-8 Final	Level 2
2-A-2-1	Solicit and integrate peer feedback as appropriate to develop or refine a program.	2-AP-15	Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
2-A-7-2	Compare different algorithms that may be used to solve the same problem, but one might be faster than the other. (e.g., different algorithms solve the same problem, but one might be faster than the other). [Clarification: Students are not expected to quantify these differences.]		Removed
2-A-7-3	Provide proper attribution when code is borrowed or built upon.	2-AP-16	Incorporate existing code, media, and libraries into original programs, and give attribution.
2-A-7-4	Interpret the flow of execution of algorithms and predict their outcomes. [Clarification: Algorithms can be expressed using natural language, flow and control diagrams, comments within code, and pseudocode.]	2-CS-01	Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.
2-A-5-5	Design, develop, and present computational artifacts such as mobile applications that address social problems both independently and collaboratively.	2-AP-18	Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
2-A-5-6	Develop programs, both independently and collaboratively, that include sequences with nested loops and multiple branches. [Clarification: At this level, students may use block-based and/or text-based programming languages.]	2-AP-12	Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
2-A-5-7	Create variables that represent different types of data and manipulate their values.	2-AP-11	Create clearly named variables that represent different data types and perform operations on their values.
2-A-4-8	Define and use procedures that hide the complexity of a task and can be reused to solve similar tasks. [Clarification: Students use and modify, but do not necessarily create, procedures with parameters.]	2-AP-14	Create procedures with parameters to organize code and make it easier to reuse.
2-A-3-9	Decompose a problem into parts and create solutions for each part.	2-AP-13	Decompose problems and subproblems into parts to facilitate the design,


CSTA & ACM, (2016), *Interim CSTA K-12 Computer Science Standards*

6-8 Interim	Level 2	6-8 Final	Level 2
			implementation, and review of programs.
2-A-6-10	Use an iterative design process (e.g., define the problem, generate ideas, build, test, and improve solutions) to solve problems, both independently and collaboratively.	2-AP-10	Use flowcharts and/or pseudocode to address complex problems as algorithms.
2-C-7-11	Justify the hardware and software chosen to accomplish a task (e.g., comparison of the features of a tablet vs. desktop, selecting which sensors and platform to use in building a robot or developing a mobile app).	2-CS-02	Design projects that combine hardware and software components to collect and exchange data.
2-C-4-12	Analyze the relationship between a device's computational components and its capabilities. [Clarification: Computing Systems include not only computers, but also cars, microwaves, smartphones, traffic lights, and flash drives.]	2-IC-20	Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.
2-C-6-13	Use a systematic process to identify the source of a problem within individual and connected devices (e.g., follow a troubleshooting flow diagram, make changes to software to see if hardware will work, restart device, check connections, swap in working components).	2-CS-03	Systematically identify and fix problems with computing devices and their components.
2-D-7-14	Describe how different formats of stored data represent tradeoffs between quality and size. [Clarification: compare examples of music, text and/or image formats.]		Removed
2-D-7-15	Explain the processes used to collect, transform, and analyze data to solve a problem using computational tools (e.g., use an app or spreadsheet form to collect data, decide which data to use or ignore, and choose a visualization method.).	2-DA-08	Collect data using computational tools and transform the data to make it more useful and reliable.
2-D-5-16	Revise computational models to more accurately reflect real-world systems (e.g., ecosystems, epidemics, spread of ideas).	2-AP-17	Systematically test and refine programs using a range of test cases.
2-D-4-17	Represent data using different encoding schemes (e.g., binary, Unicode, Morse code, shorthand, student-created codes).	2-DA-07	Represent data using multiple encoding schemes.

CSTA & ACM, (2016), *Interim CSTA K-12 Computer Science Standards*

6-8 Interim	Level 2	6-8 Final	Level 2
2-I-7-18	Summarize negative and positive impacts of using data and information to categorize people, predict behavior, and make recommendations based on those predictions (e.g., customizing search results or targeted advertising, based on previous browsing history, can save search time and limit options at the same time).	2-IC-21	Discuss issues of bias and accessibility in the design of existing technologies.
2-I-7-19	Explain how computer science fosters innovation and enhances nearly all careers and disciplines.	2-IC-22	Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.
2-I-1-20	Provide examples of how computational artifacts and devices impact health and wellbeing, both positively and negatively.		Removed
2-I-1-21	Describe ways in which the Internet impacts global communication and collaborating.		Removed
2-I-1-22	Describe ethical issues that relate to computing devices and networks (e.g., equity of access, security and plagiarism), hacking, intellectual property, copyright, Creative Commons licensing.	2-NI-05	Explain how physical and digital security measures protect electronic information.
2-I-6-23	Redesign a computational artifact to remove barriers to universal access (e.g., using captions on images, high contrast colors, and/or larger font sizes).		
2-N-7-24	Summarize security risks associated with weak passwords, lack of encryption, insecure transactions, and persistence of data.	2-IC-23	Describe tradeoffs between allowing information to be public and keeping information private and secure.
2-N-4-25	Simulate how information is transmitted as packets through multiple devices over the Internet and Networks.	2-NI-04	Model the role of protocols in transmitting data across networks and the Internet.
	New	2-NI-06	Apply multiple methods of encryption to model the secure transmission of information.
	New	2-DA-09	Refine computational models based on the data they have generated.
	New	2-AP-19	Document programs in order to make them easier to follow, test, and debug.

CSTA & ACM, (2016), *Interim CSTA K-12 Computer Science Standards*

6-8 CSTA Standards

6–8	Level 2
2-CS-01	Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.
2-CS-02	Design projects that combine hardware and software components to collect and exchange data.
2-CS-03	Systematically identify and fix problems with computing devices and their components.
2-NI-04	Model the role of protocols in transmitting data across networks and the Internet.
2-NI-05	Explain how physical and digital security measures protect electronic information.
2-NI-06	Apply multiple methods of encryption to model the secure transmission of information.
2-DA-07	Represent data using multiple encoding schemes.
2-DA-08	Collect data using computational tools and transform the data to make it more useful and reliable.
2-DA-09	Refine computational models based on the data they have generated.
2-AP-10	Use flowcharts and/or pseudocode to address complex problems as algorithms.
2-AP-11	Create clearly named variables that represent different data types and perform operations on their values.
2-AP-12	Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
2-AP-13	Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
2-AP-14	Create procedures with parameters to organize code and make it easier to reuse.
2-AP-15	Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
2-AP-16	Incorporate existing code, media, and libraries into original programs, and give attribution.
2-AP-17	Systematically test and refine programs using a range of test cases.
2-AP-18	Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
2-AP-19	Document programs in order to make them easier to follow, test, and debug.
2-IC-20	Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.
2-IC-21	Discuss issues of bias and accessibility in the design of existing technologies.
2-IC-22	Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.
2-IC-23	Describe tradeoffs between allowing information to be public and keeping information private and secure.

CSTA & ACM, (2016), *Interim CSTA K-12 Computer Science Standards*