Diocese of Venice

Standards for Technology/Computer Science

Kindergarten-12th Grade

Basic Principles Underlying All Standards to be Used for the Planning of Curriculum for the Diocese of Venice

Basic principles which inform all Catholic education in the Schools of the Diocese of Venice are:

- All knowledge, in some way, reflects God's Truth, Beauty and Goodness.
- Curriculum and instruction enable deeper incorporation of the children into the Church, the formation of community within the school; and respect for the uniqueness and dignity of each person as created in the image of God.
- Education fosters growth in Christian virtue and contributes to development and formation of the whole person in light of his/her ultimate end and the good of the society of which he/she is a member.
- Each subject is to be examined in the context of the Catholic faith and is to be illuminated by Gospel values.
- Learning and formation occur in the Catholic school without separation as does the development of each student on both the natural and supernatural levels.
- Curriculum and instruction seeks to promote a synthesis of faith, life and culture and to form students as disciples of Jesus.

Gifts of CHRIST© at the Diocese of Venice Catholic Schools

The *Gifts of C.H.R.I.S.T.©* are dispositions made up of 4 Catholic virtues (humility, prudence, fortitude, and affability) and the three transcendentals (truth, beauty, and goodness) of the Catholic faith. CHRIST in the *Gifts of C.H.R.I.S.T.©* stands for **C**atholic **H**abits and **R**esponses **i**n **S**chool and on **T**eams. Through curricular and extracurricular learning experiences at a Diocese of Venice Catholic school, PK-12th grade students will learn about the 7 *Gifts of C.H.R.I.S.T.©* and how to consciously think, behave, and respond to challenging situations using the moral and ethical values and virtues of the Catholic faith.

	GOC.T	**Truth**		
		GOC.T.1	I will seek to understand the truth in all situations.	
		GOC.T.2	I will be truthful in the way I act, and respond, and accept truth around me.	
	GOC.B	**Beauty**		
		GOC.B.1	I will seek beauty in all things and in all situations.	
		GOC.B.2	I will find the beauty of God's creation and wonders around me.	
	GOC.G	**Goodness**		
		GOC.G.1	I will exhibit goodness as God's example during my daily life.	
		GOC.G.2	I will seek goodness in learning situations and remember to see God's likeness and goodness even in my opponents.	
	GOC.A	**Affability**		
		GOC.A.1	I can be kind and treat others the way I would want to be treated.	
		GOC.A.2	I can treat others with respect, and I can be approachable and friendly.	
	GOC.H	**Humility**		
		GOC.H.1	I embrace the wisdom and knowledge that my mentors and teachers have to offer, and I understand that I can learn from their knowledge and experiences.	
		GOC.H.2	I am humbled with the knowledge and talents I have, and I understand that I can always learn more.	
	GOC.P	**Prudence**		
		GOC.P.1	I embrace learning from others, and I understand the importance of sharing my knowledge and talents by teaching others as well.	
		GOC.P.2	I can be prudent in making ethical decisions by using my Catholic faith and values as a moral compass.	
	COG.F	**Fortitude**		
		COG.F.1	I can do all things through Christ who strengthens me.	
		COG.F.2	I will seek courage in difficult situations in pursuit of what is true, beautiful, and good.	

K-2		
SC.K2.CS-CC: *K-2 Computer Science - Communication and Collaboration*		
	SC.K2.CS-CC.1: *K-2 Communication and Collaboration*	
	SC.K2.CS-CC.1.1	Identify a variety of digital tools used for communication and collaboration (e.g., online library catalogs and databases).
	SC.K2.CS-CC.1.2	Conduct basic keyword searches, and exchange information and feedback with teachers and other students (e.g., e-mail and text messaging).
	SC.K2.CS-CC.1.3	Collaborate and cooperate with peers, teachers, and others using technology to solve problems.
	SC.K2.CS-CC.1.4	Provide and accept constructive criticism on a collaborative project.
SC.K2.CS-CS: *K-2 K-2 Computer Science - Communication Systems and Computer*		
	SC.K2.CS-CS.1: *K-2 Modeling and simulations*	
	SC.K2.CS-CS.1.1	Define simulation and identify the concepts illustrated by a simple simulation (e.g., growth, human health, and the butterfly life cycle).
	SC.K2.CS-CS.1.2	Describe how models and simulations can be used to solve real-world issues in science and engineering.
	SC.K2.CS-CS.1.3	Describe how models represent a real-life system (e.g., globe or map).
	SC.K2.CS-CS.1.4	Solve questions individually and collaboratively using models.
	SC.K2.CS-CS.2: *K-2 Problem solving and algorithms*	
	SC.K2.CS-CS.2.1	Arrange or sort information into useful order, such as sorting students by birth date, with or without technology.
	SC.K2.CS-CS.2.2	Solve age-appropriate problems (e.g., puzzles and logical thinking programs) with or without technology (i.e., computational thinking).
	SC.K2.CS-CS.2.3	Solve real life issues in science and engineering using computational thinking.
	SC.K2.CS-CS.2.4	Define an algorithm as a sequence of defined steps.
	SC.K2.CS-CS.2.5	Create a simple algorithm, individually and collaboratively, without using computers to complete the task (e.g., making a sandwich, getting ready for school).

	SC.K2.CS-CS.2.6	Illustrate thoughts, ideas, and stories in a step-by-step manner using writing tools, digital cameras, and drawing tools.
	SC.K2.CS-CS.2.7	Develop and present an algorithm using tangible materials.
	SC.K2.CS-CS.2.8	Gather and organize information using concept-mapping tools.
SC.K2.CS-CS.3: *K-2 Digital tools*		
	SC.K2.CS-CS.3.1	Create a digital artifact (independently and collaboratively) that clearly expresses thoughts and ideas.
	SC.K2.CS-CS.3.2	reate, review, and revise artifacts that include text, images, and audio using digital tools.
SC.K2.CS-CS.4: *K-2 Hardware and software*		
	SC.K2.CS-CS.4.1	Recognize different kinds of computing devices in the classroom and other places (e.g., laptops, tablets, smart phones, desktops, printers).
	SC.K2.CS-CS.4.2	Recognize and operate different types of computers, applications and peripherals (e.g., use input/output devices such as a mouse, keyboard, or touch screen; find, navigate, launch a program).
	SC.K2.CS-CS.4.3	Explain that a computer program is running when a program or command is executed.
SC.K2.CS-CS.6: *K-2 Human - Computer interactions and Artificial Intelligence*		
	SC.K2.CS-CS.6.1	Identify tasks that are made easier because of computers.
SC.K2.CS-CP: *K-2 Computer Science - Computer Practices and Programing*		
SC.K2.CS-CP.1: *K-2 Data Analysis*		
	SC.K2.CS-CP.1.1	Identify different kinds of data (e.g., text, charts, graphs, numbers, pictures, audio, video, and collections of objects).
	SC.K2.CS-CP.1.2	Collect and manipulate data using a variety of computing methods (e.g., sorting, totaling, and averaging).
	SC.K2.CS-CP.1.3	Propose a solution to a problem or question based on an analysis of the data and critical thinking, individually and collaboratively.
	SC.K2.CS-CP.1.4	Create data visualizations (e.g., charts and infographics), individually and collaboratively.
SC.K2.CS-CP.2: *K-2 Computer programming basics*		
	SC.K2.CS-CP.2.1	Define a computer program as a set of commands created by people to do something.
	SC.K2.CS-CP.2.2	Perform a simple task (e.g., making a sandwich and brushing teeth) breaking it into small steps.

	SC.K2.CS-CP.2.3	Explain that computers only follow the program's instructions.
	SC.K2.CS-CP.2.4	Construct a simple program using tools that do not require a textual programming language (e.g. block-based programming language).
SC.K2.CS-CP.3: *K-2 Programming applications*		
	SC.K2.CS-CP.3.1	Create developmentally appropriate multimedia products with support from teachers, family members, or student partners.
	SC.K2.CS-CP.3.2	Prepare a simple presentation of digital products and applications.
SC.K2.CS-PC: *K-2 Computer Science - Personal, Community, Global and Ethical Impact*		
SC.K2.CS-PC.1: *K-2 Responsible use of technology and information*		
	SC.K2.CS-PC.1.1	Demonstrate proper care for electronic devices (e.g., handling devices carefully, logging off or shutting down correctly, and keeping devices away from water/food).
	SC.K2.CS-PC.1.2	Describe the attributes of a good digital citizen: one who protects private information, balances time online, reports cyberbullying, and recognizes inappropriate content/contact.
	SC.K2.CS-PC.1.3	Identify safe and unsafe examples of online communications.
	SC.K2.CS-PC.1.4	Explain that a password helps protect the privacy of information.
SC.K2.CS-PC.2: *K-2 The impact of computing resources on local and global society*		
	SC.K2.CS-PC.2.1	Identify and describe how people use many types of technologies in their daily work and personal lives.
	SC.K2.CS-PC.2.2	Communicate about technology using developmentally appropriate terminology.
	SC.K2.CS-PC.2.3	Recognize that people use computing technology in the workplace to perform many important tasks and functions.
SC.K2.CS-PC.4: *K-2 Security, privacy, information, sharing, ownership, licensure and copyright.*		
	SC.K2.CS-PC.4.1	Explain that some information is private and should not be shared online.
K-2.1A.CSTA: *Computer Science Teachers Association Standards - Level 1 A: Grades K-2 (Ages 5-7)*		
K-2.1A.CS: *Computing Systems (Grades K-2)*		
	K-2.1A-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.
	K-2.1A-CS-02	Use appropriate terminology in identifying and describing the function of common physical components of computing systems (hardware).

	K-2.1A-CS-03	Describe basic hardware and software problems using accurate terminology.
K-2.1A.NI: *Networks & the Internet (Grades K-2)*		
	K-2.1A-NI-04	Explain what passwords are and why we use them, and use strong passwords to protect devices and information from unauthorized access.
K-2.1A.DA: *Data & Analysis (Grades K-2)*		
	K-2.1A-DA-05	Store, copy, search, retrieve, modify, and delete information using a computing device and define the information stored as data.
	K-2.1A-DA-06	Collect and present the same data in various visual formats.
	K-2.1A-DA-07	Identify and describe patterns in data visualizations, such as charts or graphs, to make predictions.
K-2.1A.AP: *Algorithms & Programming (Grades K-2)*		
	K-2.1A-AP-08	Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.
	K-2.1A-AP-09	Model the way programs store and manipulate data by using numbers or other symbols to represent information.
	K-2.1A-AP-10	Develop programs with sequences and simple loops, to express ideas or address a problem.
	K-2.1A-AP-11	Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.
	K-2.1A-AP-12	Develop plans that describe a program's sequence of events, goals, and expected outcomes.
	K-2.1A-AP-13	Give attribution when using the ideas and creations of others while developing programs.
	K-2.1A-AP-14	Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.
	K-2.1A-AP-15	Using correct terminology, describe steps taken and choices made during the iterative process of program development.
K-2.1A.IC: *Impacts of Computing (Grades K-2)*		
	K-2.1A-IC-16	Compare how people live and work before and after the implementation or adoption of new computing technology.
	K-2.1A-IC-17	Work respectfully and responsibly with others online.
	K-2.1A-IC-18	Keep login information private, and log off of devices appropriately.

3-5		
SC.35.CS-CC: *3-5 Computer Science - Communication and Collaboration*		
	SC.35.CS-CC.1: *3-5 Communication and collaboration*	
	SC.35.CS-CC.1.1	Identify technology tools for individual and collaborative data collection, writing, communication, and publishing activities.
	SC.35.CS-CC.1.2	Describe key ideas and details while working individually or collaboratively using digital tools and media-rich resources in a way that informs, persuades, and/or entertains.
	SC.35.CS-CC.1.3	Identify ways that technology can foster teamwork, and collaboration can support problem solving and innovation.
	SC.35.CS-CC.1.4	Describe how collaborating with others can be beneficial to a digital project.
	SC.35.CS-CC.1.5	Explain that providing and receiving feedback from others can improve performance and outcomes for collaborative digital projects.
SC.35.CS-CS: *3-5 Computer Science - Communication Systems and Computing*		
	SC.35.CS-CS.1: *3-5 Modeling and simulations*	
	SC.35.CS-CS.1.1	Identify the concepts illustrated by a simulation (e.g., ecosystem, predator/prey, and invasive species).
	SC.35.CS-CS.1.2	Describe how models and simulations can be used to solve real-world issues in science and engineering.
	SC.35.CS-CS.1.3	Answer a question, individually and collaboratively, using data from a simulation.
	SC.35.CS-CS.1.4	Create a simple model of a system (e.g., flower or solar system) and explain what the model shows and does not show.
	SC.35.CS-CS.2: *3-5 Problem solving and algorithms*	
	SC.35.CS-CS.2.1	Solve age-appropriate problems using information organized using digital graphic organizers (e.g., concept maps and Venn-diagrams).
	SC.35.CS-CS.2.2	Describe how computational thinking can be used to solve real life issues in science and engineering.
	SC.35.CS-CS.2.3	Explain the process of arranging or sorting information into useful order as well as the purpose for doing so.
	SC.35.CS-CS.2.4	Solve real-world problems in science and engineering using computational thinking skills.

	SC.35.CS-CS.2.5	Explain that there are several possible algorithms for searching within a dataset (such as finding a specific word in a word list or card in a deck of cards).
	SC.35.CS-CS.2.6	Write an algorithm to solve a grade-level appropriate problem (e.g., move a character through a maze, instruct a character to draw a specific shape, have a character start, repeat or end activity as required or upon a specific event), individually or collaboratively.
	SC.35.CS-CS.2.7	Identify and correct logical errors in algorithms; written, mapped, live action, or digital.
	SC.35.CS-CS.2.8	Systematically test and identify logical errors in algorithms.
	SC.35.CS-CS.2.9	Explain how to correct logical errors in algorithms; written, mapped, live action, or digital.
	SC.35.CS-CS.3: *3-5 Digital tools*	
	SC.35.CS-CS.3.1	Manipulate and publish multimedia artifacts using digital tools (local and online).
	SC.35.CS-CS.3.2	Create an artifact (independently and collaboratively) that answers a research question clearly communicating thoughts and ideas.
	SC.35.CS-CS.4: *3-5 Hardware and software*	
	SC.35.CS-CS.4.1	Identify the basic components of a computer (e.g., monitor, keyboard, mouse, controller, speakers).
	SC.35.CS-CS.4.2	Describe the function and purpose of various input/output devices and peripherals (e.g., monitor, screen, keyboard, controller, speakers).
	SC.35.CS-CS.4.3	Compare and contrast hardware and software.
	SC.35.CS-CS.4.4	Identify and solve simple hardware and software problems that may occur during everyday use (e.g., power, connections, application window or toolbar).
	SC.35.CS-CS.6: *3-5 Human - Computer interactions and Artificial Intelligence*	
	SC.35.CS-CS.6.1	Describe how hardware applications (e.g., Global Positioning System (GPS) navigation for driving directions, text-to-speech translation, and language translation) can enable everyone to do things they could not do otherwise.
	SC.35.CS-CS.6.2	Compare and contrast human and computer performance on similar tasks (e.g., sorting alphabetically or finding a path across a cluttered room) to understand which is best suited to the task.
	SC.35.CS-CS.6.3	Explain that computers model intelligent behavior (as found in robotics, speech and language recognition, and computer animation).
SC.35.CS-CP: *3-5 Computer Science - Computer Practices and Programming*		
	SC.35.CS-CP.1: *3-5 Data analysis*	

	SC.35.CS-CP.1.1	Explain that searches may be enhanced by using Boolean logic (e.g., using "not", "or", "and").
	SC.35.CS-CP.1.2	Identify and describe examples of databases from everyday life (e.g., library catalogs, school records, telephone directories, and contact lists).
	SC.35.CS-CP.1.3	Identify, research, and collect a data set on a topic, issue, problem, or question using age-appropriate technologies.
	SC.35.CS-CP.1.4	Collect, organize, graph, and analyze data to answer a question using a database or spreadsheet.
	SC.35.CS-CP.2: *3-5 Computer programming basics*	
	SC.35.CS-CP.2.1	Perform keyboarding skills for communication and the input of data and information.
	SC.35.CS-CP.2.2	Create, test, and modify a program in a graphical environment (e.g., block-based visual programming language), individually and collaboratively.
	SC.35.CS-CP.2.3	Create a program using arithmetic operators, conditionals, and repetition in programs.
	SC.35.CS-CP.2.4	Explain that programs need known initial conditions (e.g., set initial score to zero in a game, initialize variables, or initial values set by hardware input).
	SC.35.CS-CP.2.5	Detect and correct program errors, including those involving arithmetic operators, conditionals, and repetition, using interactive debugging.
	SC.35.CS-CP.3: *3-5 Programming applications*	
	SC.35.CS-CP.3.1	Write, communicate and publish activities using technology tools.
	SC.35.CS-CP.3.2	Present digitally created products, either individually and collaboratively, where a topic, concept, or skill is carefully analyzed or thoughtfully explored.
SC.35.CS-PC: *3-5 Computer Science - Personal, Community, Global, and Ethical Impact*		
	SC.35.CS-PC.1: *3-5 Responsible use of technology and information*	
	SC.35.CS-PC.1.1	Identify appropriate and inappropriate uses of technology when posting to social media, sending e-mail, and browsing the Internet.
	SC.35.CS-PC.1.2	Describe responsible uses of modern communication media and devices.
	SC.35.CS-PC.1.3	Explain the proper use and operation of security technologies (e.g., passwords, virus protection software, spam filters, pop-up blockers, and cookies).
	SC.35.CS-PC.1.4	Define plagiarism and understand the impacts of plagiarized materials.

	SC.35.CS-PC.2: *3-5 The impact of computing resources on local and global society*	
	SC.35.CS-PC.2.1	Explain how computers and computing devices are used to communicate with others on a daily basis.
	SC.35.CS-PC.2.2	Describe types of cyberbullying and explain what actions should be taken if students are either victims or witnesses of these behaviors.
	SC.35.CS-PC.2.3	Identify the legal and social consequences of cyberbullying/harassment in social media.
	SC.35.CS-PC.2.4	Explain how access to technology helps empower individuals and groups (e.g., gives them access to information, the ability to communicate with others around the world, and allows them to buy and sell things).
	SC.35.CS-PC.2.5	Identify ways in which people with special needs access and use adaptive technology.
	SC.35.CS-PC.2.6	Communicate about technology using appropriate terminology.
	SC.35.CS-PC.2.7	Identify and describe how computing knowledge is essential to performing important tasks and functions.
	SC.35.CS-PC.3: *3-5 Evaluation of digital information resources*	
	SC.35.CS-PC.3.1	Identify digital information resources used to answer research questions (e.g., online library catalog, online encyclopedias, databases, and websites).
	SC.35.CS-PC.3.2	Gather, organize, and analyze information from digital resources.
	SC.35.CS-PC.3.3	Compare digital resources for accuracy, relevancy, and appropriateness.
	SC.35.CS-PC.4: *3-5 Security, privacy, information, sharing, ownership, licensure and copyright*	
	SC.35.CS-PC.4.1	Describe the difference between digital artifacts that are open or free and those that are protected by copyright.
	SC.35.CS-PC.4.2	Explain fair use for using copyrighted materials (e.g., images, music, video, and text).
	SC.35.CS-PC.4.3	Describe the purpose of copyright and the possible consequences for inappropriate use of digital materials that are protected by copyright.
	SC.35.CS-PC.4.4	Describe the threats to safe and efficient use of devices (e.g., SPAM, spyware, phishing, and viruses) associated with various forms of technology use (e.g., downloading and executing software programs, following hyperlinks, and opening files).
3-5.1A.CSTA: *Computer Science Teachers Association Standards - Level 1 B: Grades 3-5 (Ages 8-11)*		
	3-5.1A.CS: *Computing Systems (Grades 3-5)*	
	3-5.1B-CS-01	Describe how internal and external parts of computing devices function to form a system.

	3-5.1B-CS-02	Model how computer hardware and software work together as a system to accomplish tasks.
	3-5.1B-CS-03	Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies.
3-5.1A.NI: *Networks & the Internet (Grades 3-5)*		
	3-5.1B-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.
	3-5.1B-NI-05	Discuss real-world cybersecurity problems and how personal information can be protected.
3-5.1A.DA: *Data & Analysis (Grades 3-5)*		
	3-5.1B-DA-06	Organize and present collected data visually to highlight relationships and support a claim.
	3-5.1B-DA-07	Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea.
3-5.1A.AP: *Algorithms & Programming (Grades 3-5)*		
	3-5.1B-AP-08	Compare and refine multiple algorithms for the same task and determine which is the most appropriate.
	3-5.1B-AP-09	Create programs that use variables to store and modify data.
	3-5.1B-AP-10	Create programs that include sequences, events, loops, and conditionals.
	3-5.1B-AP-11	Decompose (break down) problems into smaller, manageable subproblems to facilitate the program development process.
	3-5.1B-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.
	3-5.1B-AP-13	Use an iterative process to plan the development of a program by including others' perspectives and considering user preferences.
	3-5.1B-AP-14	Observe intellectual property rights and give appropriate attribution when creating or remixing programs.
	3-5.1B-AP-15	Test and debug (identify and fix errors) a program or algorithm to ensure it runs as intended.
	3-5.1B-AP-16	Take on varying roles, with teacher guidance, when collaborating with peers during the design, implementation, and review stages of program development.
	3-5.1B-AP-17	Describe choices made during program development using code comments, presentations, and demonstrations.
3-5.1A.IC: *Impacts of Computing (Grades 3-5)*		

	3-5.1B-IC-18	Discuss computing technologies that have changed the world, and express how those technologies influence, and are influenced by, cultural practices.
	3-5.1B-IC-19	Brainstorm ways to improve the accessibility and usability of technology products for the diverse needs and wants of users.
	3-5.1B-IC-20	Seek diverse perspectives for the purpose of improving computational artifacts.
	3-5.1B-IC-21	Use public domain or creative commons media, and refrain from copying or using material created by others without permission.

Middle School		
SC.68.CS-CC: *6-8 Computer Science - Communication and Collaboration*		
	SC.68.CS-CC.1: *6-8 Communication and collaboration*	
	SC.68.CS-CC.1.1	Demonstrate an ability to communicate appropriately through various online tools.
	SC.68.CS-CC.1.2	Apply productivity and or multimedia tools for local and global group collaboration.
	SC.68.CS-CC.1.3	Design, develop, and publish a collaborative digital product using a variety of digital tools and media-rich resources that demonstrate and communicate concepts to inform, persuade, and/or entertain.
	SC.68.CS-CS	6-8 Computer Science - Communication Systems and Computing
	SC.68.CS-CS.1: *6-8 Modeling and simulations*	
	SC.68.CS-CS.1.1	Examine connections between elements of mathematics and computer science including binary numbers, logic, sets, and functions.
	SC.68.CS-CS.1.2	Create or modify and use a simulation to analyze and illustrate a concept in depth (i.e., use a simulation to illustrate a genetic variation), individually and collaboratively.
	SC.68.CS-CS.1.3	Evaluate what kinds of real-world problems can be solved using modeling and simulation.
	SC.68.CS-CS.1.4	Interact with content-specific models and simulations to support learning, research and problem solving (e.g., immigration, international trade, invasive species).
	SC.68.CS-CS.2: *6-8 Problem solving and algorithms*	
	SC.68.CS-CS.2.1	Create, modify, and use a database (e.g., define field formats, adding new records, manipulate data) to analyze data and propose solutions for a task/problem, individually and collaboratively.
	SC.68.CS-CS.2.2	Solve real-life issues in science and engineering (i.e., generalize a solution to open-ended problems) using computational thinking skills.

	SC.68.CS-CS.2.3	Perform a variety of operations such as sorting, filtering, and searching in a database.
	SC.68.CS-CS.2.4	Organize and display information in a variety of ways such as number formats (e.g., scientific notation, percentages, and exponents), charts, tables and graphs.
	SC.68.CS-CS.2.5	Decompose a problem and create a function for one of its parts at a time (e.g., video game, robot obstacle course, making dinner), individually and collaboratively.
	SC.68.CS-CS.2.6	Create a program that implements an algorithm to achieve a given goal, individually and collaboratively.
	SC.68.CS-CS.2.7	Design solutions that use repetition and two-way selection (e.g., for, while, if/else).
	SC.68.CS-CS.2.8	Recognize that boundaries need to be taken into account for an algorithm to produce correct results.
	SC.68.CS-CS.2.9	Identify simple data types and data structures.
	SC.68.CS-CS.2.10	Recognize that more than one algorithm can solve a given problem.
	SC.68.CS-CS.2.11	Predict outputs while showing an understanding of inputs.
	SC.68.CS-CS.2.12	Select the 'best' algorithm based on a given criteria (e.g., time, resource, and accessibility) to solve a problem, individually and collaboratively.
	SC.68.CS-CS.2.13	Explore a problem domain using iterative development and debugging.
	SC.68.CS-CS.2.14	Perform program tracing to predict the behavior of programs.
	SC.68.CS-CS.3: *6-8 Digital tools*	
	SC.68.CS-CS.3.1	Explain why different file types exist (e.g., formats for word processing, images, music, and three-dimensional drawings).
	SC.68.CS-CS.3.2	Identify the kinds of content associated with different file types.
	SC.68.CS-CS.3.3	Integrate information from multiple file formats into a single artifact.
	SC.68.CS-CS.4: *6-8 Hardware and software*	
	SC.68.CS-CS.4.1	Identify and describe the function of the main internal parts of a basic computing device (e.g., motherboard, hard drive, Central Processing Unit -CPU).

	SC.68.CS-CS.4.2	Describe the main functions of an operating system and explain how an operating system provides user and system services (e.g., user interface, IO device management, task management).
	SC.68.CS-CS.4.3	Describe the relationships between hardware and software (e.g., BIOS, operating systems and firmware).
	SC.68.CS-CS.4.4	Identify and describe the use of sensors, actuators, and control systems in an embodied system (e.g., a robot, an e-textile, installation art, and a smart room).
	SC.68.CS-CS.4.5	Evaluate a hardware or software problem and construct the steps involved in diagnosing and solving the problem (e.g., power, connections, application window or toolbar, cables, ports, network resources, video, and sound).
	SC.68.CS-CS.4.6	Describe the essential characteristics of a software artifact.
	SC.68.CS-CS.4.7	Describe the major components and functions of computer systems and networks.
	SC.68.CS-CS.4.8	Identify software used to support specialized forms of human-computer interaction.
SC.68.CS-CS.5: *6-8 Network systems*		
	SC.68.CS-CS.5.1	Describe how information, both text and non-text, is translated and communicated between digital computers over a computer network.
	SC.68.CS-CS.5.2	Explain the difference between physical (wired), local area wireless, and mobile networks.
	SC.68.CS-CS.5.3	Identify the major components of a network.
SC.68.CS-CS.6: *6-8 Human - Computer interactions and Artificial Intelligence*		
	SC.68.CS-CS.6.1	Explain why some tasks can be accomplished more easily by computers.
	SC.68.CS-CS.6.2	Describe how humans and machines interact to accomplish tasks that cannot be accomplished by either alone.
	SC.68.CS-CS.6.3	Identify novel ways humans interact with computers, including software, probes, sensors, and handheld devices.
	SC.68.CS-CS.6.4	Describe ways in which computers use models of intelligent behavior (e.g., robot motion, speech and language understanding, and computer vision).
	SC.68.CS-CS.6.5	Identify factors that distinguish humans from machines.
	SC.68.CS-CS.6.6	Design and demonstrate the use of a device (e.g., robot, e-textile) to accomplish a task, individually and collaboratively.
SC.68.CS-CP: *6-8 Computer Science - Computer Practices and Programming*		

	SC.68.CS-CP.1: *6-8 Data analysis*	
	SC.68.CS-CP.1.1	Define parameters for individual and collaborative projects using Boolean logic (e.g., using "not", "or", "and").
	SC.68.CS-CP.1.2	Select and use data-collection technology (e.g., probes, handheld devices, geographic mapping systems and output from multiple runs of a computer program) to gather, view, organize, analyze, and report results for content-related problems, individually and collaboratively.
	SC.68.CS-CP.2: *6-8 Computer programming basics*	
	SC.68.CS-CP.2.1	Develop problem solutions using visual representations of problem states, structures and data.
	SC.68.CS-CP.2.2	Evaluate the logical flow of a step-by-step program by acting it out through computer-free activities.
	SC.68.CS-CP.2.3	Develop problem solutions using a block programming language, including all of the following: looping behavior, conditional statements, expressions, variables, and functions.
	SC.68.CS-CP.2.4	Develop problem solutions using a programming language, including all of the following: looping behavior, conditional statements, expressions, variables, and functions.
	SC.68.CS-CP.3: *6-8 Programming applications*	
	SC.68.CS-CP.3.1	Select appropriate tools and technology resources to accomplish a variety of tasks and solve problems.
	SC.68.CS-CP.3.2	Create online content (e.g., webpage, blog, digital portfolio, multimedia), using advanced design tools.
	SC.68.CS-CP.3.3	Create an artifact (independently and collaboratively) that answers a research question and communicates results and conclusions.
SC.68.CS-PC: *6-8 Computer Science - Personal, Community, Global, and Ethical Impact*		
	SC.68.CS-PC.1: *6-8 Responsible use of technology and information*	
	SC.68.CS-PC.1.1	Recognize and describe legal and ethical behaviors when using information and technology and describe the consequences of misuse.
	SC.68.CS-PC.1.2	Describe and use safe and appropriate practices when participating in online communities (e.g., discussion groups, blogs, and social networking sites).
	SC.68.CS-PC.1.3	Evaluate the proper use and operation of security technologies (e.g., passwords, virus protection software, spam filters, pop-up blockers, and cookies).
	SC.68.CS-PC.1.4	Recognize the impacts and consequences of plagiarism on the development of creative works, projects, publications and online content.
	SC.68.CS-PC.2: *6-8 The impact of computing resources on local and global society*	

	SC.68.CS-PC.2.1	Analyze the positive and negative impacts of computing, social networking and web technologies on human culture.
	SC.68.CS-PC.2.2	Explain the possible consequences of cyberbullying and inappropriate use of social media on personal life and society.
	SC.68.CS-PC.2.3	Describe the influence of access to information technologies over time and the effects those changes have had on education, the workplace, and the global society.
	SC.68.CS-PC.2.4	Describe how the unequal net-neutrality and distribution of computing resources in a global economy raises issues of equity, access, and power.
	SC.68.CS-PC.2.5	Describe ways in which adaptive technologies can assist users with special needs to function in their daily lives.
	SC.68.CS-PC.2.6	Identify and discuss the technology skills needed in the workplace.
	SC.68.CS-PC.2.7	Interpret writings and/or communications which use developmentally appropriate terminology.
	SC.68.CS-PC.2.8	Identify interdisciplinary careers that are enhanced by computer science.
SC.68.CS-PC.3: *6-8 Evaluation of digital information resources*		
	SC.68.CS-PC.3.1	Answer research questions using digital information resources.
	SC.68.CS-PC.3.2	Analyze how media and technology can be used to distort, exaggerate, or misrepresent information.
	SC.68.CS-PC.3.3	Describe strategies for determining the reliability of resources or information on the Internet.
	SC.68.CS-PC.3.4	Identify peer reviewed resources and understand the need for peer review.
	SC.68.CS-PC.3.5	Identify resources such as city, state, and federal government websites and explain that these resources can be used for communication between citizens and government.
SC.68.CS-PC.4: *6-8 Security, privacy, information, sharing, ownership, licensure and copyright*		
	SC.68.CS-PC.4.1	Explain the guidelines for the fair use of downloading, sharing or modifying of digital materials.
	SC.68.CS-PC.4.2	Explain how copyright law and licensing protect the owner of intellectual properties.
	SC.68.CS-PC.4.3	Explain the possible consequences of violating intellectual property law.
	SC.68.CS-PC.4.4	Identify threats and actions that protect devices from viruses, intrusion, vandalism, and other malicious activities.
	SC.68.CS-PC.4.5	Demonstrate compliance with the school's Acceptable Use Policy.

	SC.68.CS-PC.4.6	Generate text and non-text citations using digital citation tool.
6-8.2.CSTA: *Computer Science Teachers Association Standards - Level 2: Grades 6-8 (Ages 11-14)*		
6-8.2.CS: *Computing Systems (Grades 6-8)*		
	6-8.2-CS-01	Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.
	6-8.2-CS-02	Design projects that combine hardware and software components to collect and exchange data.
	6-8.2-CS-03	Systematically identify and fix problems with computing devices and their components.
6-8.2.NI: *Networks & the Internet (Grades 6-8)*		
	6-8.2-NI-04	Model the role of protocols in transmitting data across networks and the Internet.
	6-8.2-NI-05	Explain how physical and digital security measures protect electronic information.
	6-8.2-NI-06	Apply multiple methods of encryption to model the secure transmission of information.
6-8.2.DA: *Data & Analysis (Grades 6-8)*		
	6-8.2-DA-07	Represent data using multiple encoding schemes.
	6-8.2-DA-08	Collect data using computational tools and transform the data to make it more useful and reliable.
	6-8.2-DA-09	Refine computational models based on the data they have generated.
6-8.2.AP: *Algorithms & Programming (Grades 6-8)*		
	6-8.2-AP-10	Use flowcharts and/or pseudocode to address complex problems as algorithms.
	6-8.2-AP-11	Create clearly named variables that represent different data types and perform operations on their values.
	6-8.2-AP-12	Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
	6-8.2-AP-13	Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
	6-8.2-AP-14	Create procedures with parameters to organize code and make it easier to reuse.
	6-8.2-AP-15	Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
	6-8.2-AP-16	Incorporate existing code, media, and libraries into original programs, and give attribution.
	6-8.2-AP-17	Systematically test and refine programs using a range of test cases.

	6-8.2-AP-18	Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
	6-8.2-AP-19	Document programs in order to make them easier to follow, test, and debug.
6-8.2.IC: *Impacts of Computing (Grades 6-8)*		
	6-8.2-IC-20	Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.
	6-8.2-IC-21	Discuss issues of bias and accessibility in the design of existing technologies.
	6-8.2-IC-22	Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.
	6-8.2-IC-23	Describe tradeoffs between allowing information to be public and keeping information private and secure.

High School		
SC.912.CS-CC: *9-12 Computer Science - Communication and Collaboration*		
SC.912.CS-CC.1: *9-12 Communication and collaboration*		
	SC.912.CS-CC.1.1	Evaluate modes of communication and collaboration.
	SC.912.CS-CC.1.2	Select appropriate tools within a project environment to communicate with project team members.
	SC.912.CS-CC.1.3	Collect, analyze, and present information using a variety of computing devices (e.g., probes, sensors, and handheld devices).
	SC.912.CS-CC.1.4	Develop a collaborative digital product using collaboration tools (e.g., version control systems and integrated development environments).
	SC.912.CS-CC.1.5	Communicate and publish key ideas and details to a variety of audiences using digital tools and media-rich resources.
	SC.912.CS-CC.1.6	Identify how collaboration influences the design and development of software artifacts.
	SC.912.CS-CC.1.7	Evaluate program designs and implementations written by others for readability and usability.
SC.912.CS-CS: *9-12 Computer Science - Communication Systems and Computing*		
SC.912.CS-CS.1: *9-12 Modeling and simulations*		
	SC.912.CS-CS.1.1	Analyze data and identify real-world patterns through modeling and simulation.
	SC.912.CS-CS.1.2	Formulate, refine, and test scientific hypotheses using models and simulations.

	SC.912.CS-CS.1.3	Explain how data analysis is used to enhance the understanding of complex natural and human systems.
	SC.912.CS-CS.1.4	Compare techniques for analyzing massive data collections.
	SC.912.CS-CS.1.5	Represent and understand natural phenomena using modeling and simulation.
SC.912.CS-CS.2: *9-12 Problem solving and algorithms*		
	SC.912.CS-CS.2.1	Explain intractable problems and understand that problems exists that are computationally unsolvable (e.g., classic intractable problems include the Towers of Hanoi and the Traveling Salesman Problem -TSP).
	SC.912.CS-CS.2.2	Describe the concept of parallel processing as a strategy to solve large problems.
	SC.912.CS-CS.2.3	Demonstrate concurrency by separating processes into threads of execution and dividing data into parallel streams.
	SC.912.CS-CS.2.4	Divide a complex problem into simpler parts by using the principle of abstraction to manage complexity (i.e., by using searching and sorting as abstractions) using predefined functions and parameters, classes, and methods.
	SC.912.CS-CS.2.5	Evaluate a classical algorithms and implement an original algorithm.
	SC.912.CS-CS.2.6	Evaluate various data types and data structures.
	SC.912.CS-CS.2.7	Explain how sequence, selection, iteration, and recursion are building blocks of algorithms.
	SC.912.CS-CS.2.8	Decompose a problem by defining new functions and classes.
	SC.912.CS-CS.2.9	Evaluate ways to characterize how well algorithms perform and that two algorithms can perform differently for the same task.
	SC.912.CS-CS.2.10	Design and implement a simple simulation algorithm to analyze, represent, and understand natural phenomena.
	SC.912.CS-CS.2.11	Evaluate algorithms by their efficiency, correctness, and clarity (e.g., by analyzing and comparing execution times, testing with multiple inputs or data sets, and by debugging).
	SC.912.CS-CS.2.12	Compare and contrast simple data structures and their uses.
	SC.912.CS-CS.2.13	Explain how automated software testing can reduce the cost of the testing effort.
	SC.912.CS-CS.2.14	Explain what tools are applied to provide automated testing environments.
SC.912.CS-CS.3: *9-12 Digital tools*		

	SC.912.CS-CS.3.1	Describe digital tools or resources to use for a real-world task based on their efficiency and effectiveness.
	SC.912.CS-CS.3.2	Evaluate different file types for different purposes (e.g., word processing, images, music, and three-dimensional drawings).
SC.912.CS-CS.4: *9-12 Hardware and software*		
	SC.912.CS-CS.4.1	Describe a software development process that is used to solve problems at different software development stages (e.g., design, coding, testing, and verification).
	SC.912.CS-CS.4.2	Describe the organization of a computer and identify its principal components by name, function, and the flow of instructions and data between components (e.g., storage devices, memory, CPU, graphics processors, IO and network ports).
	SC.912.CS-CS.4.3	Differentiate between multiple levels of hardware and software (such as CPU hardware, operating system, translation, and interpretation) that support program execution.
	SC.912.CS-CS.4.4	Evaluate various forms of input and output (e.g., IO and storage devices and digital media).
	SC.912.CS-CS.4.5	Develop and evaluate criteria for purchasing or upgrading computer system hardware (e.g., Wi-Fi, mobile devices, home and office machines).
	SC.912.CS-CS.4.6	Develop criteria for selecting appropriate hardware and software when solving a specific real-world problem (such as business, educational, personal).
	SC.912.CS-CS.4.7	Develop a software artifact (independently and collaboratively) in phases (or stages) according to a common software development methodology (e.g., Waterfall or Spiral model).
	SC.912.CS-CS.4.8	Evaluate the basic components of computer networks.
	SC.912.CS-CS.4.9	Analyze historical trends in hardware and software to assess implications on computing devices for the future (e.g., upgrades for power/energy, computation capacity, speed, size, ease of use).
SC.912.CS-CS.5: *9-12 Network systems*		
	SC.912.CS-CS.5.1	Identify and select the most appropriate file format based on trade-offs (e.g., open file formats, text, proprietary and binary formats, compression and encryption formats).
	SC.912.CS-CS.5.2	Describe the issues that impact network functionality (e.g., latency, bandwidth, firewalls and server capability).
	SC.912.CS-CS.5.3	Describe common network protocols, such as IP, TCP, SMTP, HTTP, and FTP, and how these are applied by client-server and peer-to-peer networks.
SC.912.CS-CS.6: *9-12 Human - Computer interactions and Artificial Intelligence*		
	SC.912.CS-CS.6.1	Describe the unique features of computers embedded in mobile devices and vehicles.

	SC.912.CS-CS.6.2	Describe the common physical and cognitive challenges faced by users when learning to use software and hardware.
	SC.912.CS-CS.6.3	Describe the process of designing software to support specialized forms of human-computer interaction.
	SC.912.CS-CS.6.4	Explain the notion of intelligent behavior through computer modeling and robotics.
	SC.912.CS-CS.6.5	Describe common measurements of machine intelligence (e.g., Turing test).
	SC.912.CS-CS.6.6	Describe a few of the major branches of artificial intelligence (e.g., expert systems, natural language processing, machine perception, machine learning).
	SC.912.CS-CS.6.7	Describe major applications of artificial intelligence and robotics, including, but not limited to, the medical, space, and automotive fields.
SC.912.CS-CP: *9-12 Computer Science - Computer Practices and Programming*		
	SC.912.CS-CP.1: *9-12 Data analysis*	
	SC.912.CS-CP.1.1	Evaluate effective uses of Boolean logic (e.g., using "not", "or", "and") to refine searches for individual and collaborative projects.
	SC.912.CS-CP.1.2	Perform advanced searches to locate information and/or design a data-collection approach to gather original data (e.g., qualitative interviews, surveys, prototypes, and simulations).
	SC.912.CS-CP.1.3	Analyze and manipulate data collected by a variety of data collection techniques to support a hypothesis.
	SC.912.CS-CP.1.4	Collect real-time data from sources such as simulations, scientific and robotic sensors, and device emulators, using this data to formulate strategies or algorithms to solve advanced problems.
	SC.912.CS-CP.2: *9-12 Computer programming basics*	
	SC.912.CS-CP.2.1	Explain the program execution process (by an interpreter and in CPU hardware).
	SC.912.CS-CP.2.2	Design and implement a program using global and local scope.
	SC.912.CS-CP.2.3	Implement a program using an industrial-strength integrated development environment.
	SC.912.CS-CP.2.4	Facilitate programming solutions using application programming interfaces (APIs) and libraries.
	SC.912.CS-CP.2.5	Explain the role of an API in the development of applications and the distinction between a programming language’s syntax and the API.
	SC.912.CS-CP.2.6	Describe a variety of commonly used programming languages.

	SC.912.CS-CP.2.7	Classify programming languages by paradigm and application domain (e.g., imperative, functional, and logic languages) and evaluate their application to domains such as web programming, symbolic processing and data/numerical processing.
SC.912.CS-CP.3: *9-12 Programming applications*		
	SC.912.CS-CP.3.1	Create a computational artifact, individually and collaboratively, followed by reflection, analysis, and iteration (e.g., data-set analysis program for science and engineering fair, capstone project that includes a program, term research project based on program data).
	SC.912.CS-CP.3.2	Create mobile computing applications and/or dynamic web pages through the use of a variety of design and development tools, programming languages, and mobile devices/emulators.
	SC.912.CS-PC	9-12 Computer Science - Personal, Community, Global, and Ethical Impact
SC.912.CS-PC.1: *9-12 Responsible use of technology and information*		
	SC.912.CS-PC.1.1	Compare and contrast appropriate and inappropriate social networking behaviors.
	SC.912.CS-PC.1.2	Describe and demonstrate ethical and responsible use of modern communication media and devices.
	SC.912.CS-PC.1.3	Evaluate the impacts of irresponsible use of information (e.g., plagiarism and falsification of data) on collaborative projects.
	SC.912.CS-PC.1.4	Explain the principles of cryptography by examining encryption, digital signatures, and authentication methods (e.g., explain why and how certificates are used with "https" for authentication and encryption).
	SC.912.CS-PC.1.5	Implement an encryption, digital signature, or authentication method.
	SC.912.CS-PC.1.6	Describe computer security vulnerabilities and methods of attack, and evaluate their social and economic impact on computer systems and people.
SC.912.CS-PC.2: *9-12 The impact of computing resources on local and global society*		
	SC.912.CS-PC.2.1	Describe how the Internet facilitates global communication.
	SC.912.CS-PC.2.2	Identify ways to use technology to support lifelong learning.
	SC.912.CS-PC.2.3	Discuss and analyze the impact of values and points of view that are presented in media messages (e.g., racial, gender, and political).
	SC.912.CS-PC.2.4	Analyze the positive and negative impacts of technology on popular culture and personal life.
	SC.912.CS-PC.2.5	Construct strategies to combat cyberbullying or online harassment.

	SC.912.CS-PC.2.6	Describe the impact of computing on business and commerce (e.g., automated inventory processing, financial transactions, e-commerce, virtualization, and cloud computing).
	SC.912.CS-PC.2.7	Describe how technology has changed the way people build and manage organizations and how technology impacts personal life.
	SC.912.CS-PC.2.8	Evaluate ways in which adaptive technologies may assist users with special needs.
	SC.912.CS-PC.2.9	Explain how societal and economic factors are affected by access to critical information.
	SC.912.CS-PC.2.10	Describe and evaluate the challenges (e.g., political, social, and economic) in providing equal access and distribution of technology in a global society.
	SC.912.CS-PC.2.11	Construct writings and/or communications using developmentally appropriate terminology.
	SC.912.CS-PC.2.12	Explore a variety of careers to which computing is central.
	SC.912.CS-PC.2.13	Predict future careers and the technologies that may exist based on current technology trends.
SC.912.CS-PC.3: *9-12 Evaluation of digital information resources*		
	SC.912.CS-PC.3.1	Evaluate the quality of digital resources for reliability (i.e., currency, relevancy, authority, accuracy, and purpose of digital information).
	SC.912.CS-PC.3.2	Evaluate the accuracy, relevance, comprehensiveness, appropriateness, and bias of electronic information resources.
	SC.912.CS-PC.3.3	Conduct research using peer reviewed articles, newspapers, magazine articles, and online books.
	SC.912.CS-PC.3.4	Analyze and evaluate public/government resources and describe how using these resources for communication can affect change.
SC.912.CS-PC.4: *9-12 Security, privacy, information, sharing, ownership, licensure and copyright*		
	SC.912.CS-PC.4.1	Describe how different types of software licenses (e.g., open source and proprietary licenses) can be used to share and protect intellectual property.
	SC.912.CS-PC.4.2	Explain how access to information may not include the right to distribute the information.
	SC.912.CS-PC.4.3	Describe differences between open source, freeware, and proprietary software licenses, and how they apply to different types of software.
	SC.912.CS-PC.4.4	Describe security and privacy issues that relate to computer networks.
	SC.912.CS-PC.4.5	Identify computer-related laws and analyze their impact on digital privacy, security, intellectual property, network access, contracts, and harassment.

	SC.912.CS-PC.4.6	Describe security and privacy issues that relate to computer networks including the permanency of data on the Internet, online identity, and privacy.
	SC.912.CS-PC.4.7	Evaluate and use digital citation tools to cite sources.
	SC.912.CS-PC.4.8	Describe the impact of government regulation on privacy and security.
9-10.3A.CSTA: ***Computer Science Teachers Association Standards - Level 3A: Grades 9-10 (Ages 14-16)***		
9-10.3A.CS: *Computing Systems (Grades 9-10)*		
	9-10.3A-CS-01	Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.
	9-10.3A-CS-02	Compare levels of abstraction and interactions between application software, system software, and hardware layers.
	9-10.3A-CS-03	Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
9-10.3A.NI: *Networks & the Internet (Grades 9-10)*		
	9-10.3A-NI-04	Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing.
	9-10.3A-NI-05	Give examples to illustrate how sensitive data can be affected by malware and other attacks.
	9-10.3A-NI-06	Recommend security measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.
	9-10.3A-NI-07	Compare various security measures, considering tradeoffs between the usability and security of a computing system.
	9-10.3A-NI-08	Explain tradeoffs when selecting and implementing cybersecurity recommendations.
9-10.3A.DA: *Data & Analysis (Grades 9-10)*		
	9-10.3A-DA-09	Translate between different bit representations of real-world phenomena, such as characters, numbers, and images.
	9-10.3A-DA-10	Evaluate the tradeoffs in how data elements are organized and where data is stored.
	9-10.3A-DA-11	Create interactive data visualizations using software tools to help others better understand real-world phenomena.
	9-10.3A-DA-12	Create computational models that represent the relationships among different elements of data collected from a phenomenon or process.
9-10.3A.AP: *Algorithms & Programming (Grades 9-10)*		

	9-10.3A-AP-13	Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
	9-10.3A-AP-14	Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
	9-10.3A-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.
	9-10.3A-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.
	9-10.3A-AP-17	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
	9-10.3A-AP-18	Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
	9-10.3A-AP-19	Systematically design and develop programs for broad audiences by incorporating feedback from users.
	9-10.3A-AP-20	Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.
	9-10.3A-AP-21	Evaluate and refine computational artifacts to make them more usable and accessible.
	9-10.3A-AP-22	Design and develop computational artifacts working in team roles using collaborative tools.
	9-10.3A-AP-23	Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.
	9-10.3A.IC: *Impacts of Computing (Grades 9-10)*	
	9-10.3A-IC-24	Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.
	9-10.3A-IC-25	Test and refine computational artifacts to reduce bias and equity deficits.
	9-10.3A-IC-26	Demonstrate ways a given algorithm applies to problems across disciplines.
	9-10.3A-IC-27	Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields.
	9-10.3A-IC-28	Explain the beneficial and harmful effects that intellectual property laws can have on innovation.

	9-10.3A-IC-29	Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.
	9-10.3A-IC-30	Evaluate the social and economic implications of privacy in the context of safety, law, or ethics.
11-12.3B.CSTA: *Computer Science Teachers Association Standards - Level 3B: Grades 11-12 (Ages 16-18)*		
	11-12.3B.CS: *Computing Systems (Grades 11-12)*	
	11-12.3B-CS-01	Categorize the roles of operating system software.
	11-12.3B-CS-02	Illustrate ways computing systems implement logic, input, and output through hardware components.
	11-12.3B.NI: *Networks & the Internet (Grades 11-12)*	
	11-12.3B-NI-03	Describe the issues that impact network functionality (e.g., bandwidth, load, delay, topology).
	11-12.3B-NI-04	Compare ways software developers protect devices and information from unauthorized access.
	11-12.3B.DA: *Data & Analysis (Grades 11-12)*	
	11-12.3B-DA-05	Use data analysis tools and techniques to identify patterns in data representing complex systems.
	11-12.3B-DA-06	Select data collection tools and techniques to generate data sets that support a claim or communicate information.
	11-12.3B-DA-07	Evaluate the ability of models and simulations to test and support the refinement of hypotheses.
	11-12.3B.AP: *Algorithms & Programming (Grades 11-12)*	
	11-12.3B-AP-08	Describe how artificial intelligence drives many software and physical systems.
	11-12.3B-AP-09	Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.
	11-12.3B-AP-10	Use and adapt classic algorithms to solve computational problems.
	11-12.3B-AP-11	Evaluate algorithms in terms of their efficiency, correctness, and clarity.
	11-12.3B-AP-12	Compare and contrast fundamental data structures and their uses.
	11-12.3B-AP-13	Illustrate the flow of execution of a recursive algorithm.
	11-12.3B-AP-14	Construct solutions to problems using student-created components, such as procedures, modules and/or objects.

	11-12.3B-AP-15	Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution.
	11-12.3B-AP-16	Demonstrate code reuse by creating programming solutions using libraries and APIs.
	11-12.3B-AP-17	Plan and develop programs for broad audiences using a software life cycle process.
	11-12.3B-AP-18	Explain security issues that might lead to compromised computer programs.
	11-12.3B-AP-19	Develop programs for multiple computing platforms.
	11-12.3B-AP-20	Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (code documentation) in a group software project.
	11-12.3B-AP-21	Develop and use a series of test cases to verify that a program performs according to its design specifications.
	11-12.3B-AP-22	Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).
	11-12.3B-AP-23	Evaluate key qualities of a program through a process such as a code review.
	11-12.3B-AP-24	Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems.
11-12.3B.IC: *Impacts of Computing (Grades 11-12)*		
	11-12.3B-IC-25	Evaluate computational artifacts to maximize their beneficial effects and minimize harmful effects on society.
	11-12.3B-IC-26	Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society.
	11-12.3B-IC-27	Predict how computational innovations that have revolutionized aspects of our culture might evolve.
	11-12.3B-IC-28	Debate laws and regulations that impact the development and use of software.

AP COMPUTER SCIENCE		
APCS.MOD.1: *Modularity - Some objects or concepts are so frequently represented that programmers can draw upon existing code that has already been tested, enabling them to write solutions more quickly and with a greater degree of confidence.*		
	APCS.MOD.SK: *Suggested Skills*	
	APCS.MOD.SK.1.A	Determine an appropriate program design to solve a problem or accomplish a task (not assessed).
	APCS.MOD.SK.1.B	Determine code that would be used to complete code segments.
	APCS.MOD.SK.1.C	Determine code that would be used to complete code segments.
	APCS.MOD.SK.2.A	Apply the meaning of specific operators.
	APCS.MOD.SK.2.B	Determine the result or output based on statement execution order in a code segment without method calls (other than output).
	APCS.MOD.SK.2.C	Determine the result or output based on statement execution order in a code segment without method calls (other than output).
	APCS.MOD.SK.3.A	Write program code to create objects of a class and call methods.
	APCS.MOD.SK.3.B	Write program code to define a new type by creating a class.
	APCS.MOD.SK.4.B	Identify errors in program code.
	APCS.MOD.SK.5.A	Describe the behavior of a given segment of program code.
	APCS.MOD.SK.5.B	Explain why a code segment will not compile or work as intended.
	APCS.MOD.SK.5.D	Describe the initial conditions that must be met for a program segment to work as intended or described.
	APCS.MOD.1.A: *Call system class methods to generate output to the console.*	
	APCS.MOD.1.A.1	System.out.print and System.out.println display information on the computer monitor.
	APCS.MOD.1.A.2	System.out.println moves the cursor to a new line after the information has been displayed, while System.out.print does not.
	APCS.MOD.1.B: *Explain the relationship between a class and an object.*	
	APCS.MOD.1.B.1	An object is a specific instance of a class with defined attributes.

	APCS.MOD.1.B.2	A class is the formal implementation, or blueprint, of the attributes and behaviors of an object.
APCS.MOD.1.C: *Identify, using its singnature, the correct constructor being called.*		
	APCS.MOD.1.C.1	A signature consists of the constructor name and the parameter list.
	APCS.MOD.1.C.2	The parameter list, in the header of a constructor, lists the types of the values that are passed and their variable names. These are often referred to as formal parameters.
	APCS.MOD.1.C.3	A parameter is a value that is passed into a constructor. These are often referred to as actual parameters.
	APCS.MOD.1.C.4	Constructors are said to be overloaded when there are multiple constructors with the same name but a different signature.
	APCS.MOD.1.C.5	The actual parameters passed to a constructor must be compatible with the types identified in the formal parameter list.
	APCS.MOD.1.C.6	Parameters are passed using call by value. Call by value initializes the formal parameters with copies of the actual parameters.
APCS.MOD.1.D: *For creating objects: a.) Create objects by calling constructors without parameters. b.) Create objects by calling constructors with parameters.*		
	APCS.MOD.1.D.1	Every object is created using the keyword new followed by a call to one of the class's constructors.
	APCS.MOD.1.D.2	A class contains constructors that are invoked to create objects. They have the same name as the class.
	APCS.MOD.1.D.3	Existing classes and class libraries can be utilized as appropriate to create objects.
	APCS.MOD.1.D.4	Parameters allow values to be passed to the constructor to establish the initial state of the object.
APCS.MOD.1.E: *Call non-static void methods without parameters.*		
	APCS.MOD.1.E.1	An object's behavior refers to what the object can do (or what can be done to it) and is defined by methods.
	APCS.MOD.1.E.2	Procedural abstraction allows a programmer to use a method by knowing what the method does even if they do not know how the method was written.
	APCS.MOD.1.E.3	A method signature for a method without parameters consists of the method name and an empty parameter list.

	APCS.MOD.1.E.4	A method or constructor call interrupts the sequential execution of statements, causing the program to first execute the statements in the method or constructor before continuing. Once the last statement in the method or constructor has executed or a return statement is executed, flow of control is returned to the point immediately following where the method or constructor was called.
	APCS.MOD.1.E.5	Non-static methods are called through objects of the class.
	APCS.MOD.1.E.6	The dot operator is used along with the object name to call non-static methods.
	APCS.MOD.1.E.7	Using a null reference to call a method or access an instance variable causes a NullPointerException to be thrown.
	APCS.MOD.1.F: *Call non-static void methods with parameters.*	
	APCS.MOD.1.F.1	A method signature for a method with parameters consists of the method name and the ordered list of parameter types.
	APCS.MOD.1.F.2	Values provided in the parameter list need to correspond to the order and type in the method signature.
	APCS.MOD.1.F.3	Methods are said to be overloaded when there are multiple methods with the same name but a different signature.
	APCS.MOD.1.G: *Call non-static non-void methods with or without parameters.*	
	APCS.MOD.1.G.1	Non-void methods return a value that is the same type as the return type in the signature. To use the return value when calling a non-void method, it must be stored in a variable or used as part of an expression.
	APCS.MOD.1.H: *Call static methods.*	
	APCS.MOD.1.H.1	Static methods are called using the dot operator along with the class name unless they are defined in the enclosing class.
APCS.MOD.2: *Modularity - Programmers use code to represent a physical object or nonphysical oncept, real or imagined, by defining a class based on the attributes and/or behaviors of the object or concept.*		
	APCS.MOD.2.A: *Designate access and visibility constraints to classes, data, constructors, and methods.*	
	APCS.MOD.2.A.1	The keywords public and private affect the access of classes, data, constructors, and methods.
	APCS.MOD.2.A.2	The keyword private restricts access to the declaring class, while the keyword publicallows access from classes outside the declaring class.
	APCS.MOD.2.A.3	Classes are designated public.
	APCS.MOD.2.A.4	Access to attributes should be kept internal to the class. Therefore, instance variables are designated as private.

	APCS.MOD.2.A.5	Constructors are designated public.
	APCS.MOD.2.A.6	Access to behaviors can be internal or external to the class. Therefore, methods can be designated as either public or private.
APCS.MOD.2.B: *Define instance variables for the attributes to be initiailized through the constructors of a class.*		
	APCS.MOD.2.B.1	An object's state refers to its attributes and their values at a given time and is defined by instance variables belonging to the object. This creates a "has-a" relationship between the object and its instance variables.
	APCS.MOD.2.B.2	Constructors are used to set the initial state of an object, which should include initial values for all instance variables.
	APCS.MOD.2.B.3	Constructor parameters are local variables to the constructor and provide data to initialize instance variables.
	APCS.MOD.2.B.4	When a mutable object is a constructor parameter, the instance variable should be initialized with a copy of the referenced object. In this way, the instance variable is not an alias of the original object, and methods are prevented from modifying the state of the original object.
	APCS.MOD.2.B.5	When no constructor is written, Java provides a no-argument constructor, and the instance variables are set to default values.
APCS.MOD.2.C: *Describe the functionality and use of program code through comments.*		
	APCS.MOD.2.C.1	Comments are ignored by the compiler and are not executed when the program is run.
	APCS.MOD.2.C.2	Three types of comments in Java include /* */, which generates a block of comments, //, which generates a comment on one line, and /** */, which are Javadoc comments and are used to create API documentation.
	APCS.MOD.2.C.3	A precondition is a condition that must be true just prior to the execution of a section of program code in order for the method to behave as expected. There is no expectation that the method will check to ensure preconditions are satisfied.
	APCS.MOD.2.C.4	A postcondition is a condition that must always be true after the execution of a section of program code. Postconditions describe the outcome of the execution in terms of what is being returned or the state of an object.
	APCS.MOD.2.C.5	Programmers write method code to satisfy the postconditions when preconditions are met.
APCS.MOD.2.D: *Define behaviors of an object through non-void methods without parameters written in a class.*		
	APCS.MOD.2.D.1	An accessor method allows other objects to obtain the value of instance variables or static variables.

	APCS.MOD.2.D.2	A non-void method returns a single value. Its header includes the return type in place of the keyword void.
	APCS.MOD.2.D.3	In non-void methods, a return expression compatible with the return type is evaluated, and a copy of that value is returned. This is referred to as "return by value."
	APCS.MOD.2.D.4	When the return expression is a reference to an object, a copy of that reference is returned, not a copy of the object.
	APCS.MOD.2.D.5	The return keyword is used to return the flow of control to the point immediately following where the method or constructor was called.
	APCS.MOD.2.D.6	The toString method is an overridden method that is included in classes to provide a description of a specific object. It generally includes what values are stored in the instance data of the object.
	APCS.MOD.2.D.7	If System.out.print or System.out.println is passed an object, that object's toString method is called, and the returned string is printed.
APCS.MOD.2.E: *Define behaviors of an object through void methods with or without parameters written in a class.*		
	APCS.MOD.2.E.1	A void method does not return a value. Its header contains the keyword void before the method name.
	APCS.MOD.2.E.2	A mutator (modifier) method is often a void method that changes the values of instance variables or static variable.
APCS.MOD.2.F: *Define behaviors of an object through non-void methods with parameters written in a class.*		
	APCS.MOD.2.F.1	Methods can only access the private data and methods of a parameter that is a reference to an object when the parameter is the same type as the method's enclosing class.
	APCS.MOD.2.F.2	Non-void methods with parameters receive values through parameters, use those values, and return a computed value of the specified type.
	APCS.MOD.2.F.3	It is good programming practice to not modify mutable objects that are passed as parameters unless required in the specification.
	APCS.MOD.2.F.4	When an actual parameter is a primitive value, the formal parameter is initialized with a copy of that value. Changes to the formal parameter have no effect on the corresponding actual parameter.
	APCS.MOD.2.F.5	When an actual parameter is a reference to an object, the formal parameter is initialized with a copy of that reference, not a copy of the object. If the reference is to a mutable object, the method or constructor can use this reference to alter the state of the object.
APCS.MOD.2.G: *Define behaviors of a class through static methods.*		
	APCS.MOD.2.G.1	Static methods are associated with the class, not objects of the class.

	APCS.MOD.2.G.2	Static methods include the keyword staticin the header before the method name.
	APCS.MOD.2.G.3	Static methods cannot access or change the values of instance variables.
	APCS.MOD.2.G.4	Static methods can access or change the values of static variables.
	APCS.MOD.2.G.5	Static methods do not have a this reference and are unable to use the class's instance variables or call non-static methods.
APCS.MOD.2.H: *Define the static variables that belong to the class.*		
	APCS.MOD.2.H.1	Static variables belong to the class, with all objects of a class sharing a single static variable.
	APCS.MOD.2.H.2	Static variables can be designated as either public or private and are designated with the static keyword before the variable type.
	APCS.MOD.2.H.3	Static variables are used with the class name and the dot operator, since they are associated with a class, not objects of a class.
APCS.MOD.3: *Modularity - When multiple classes contain common attributes and behaviors, programmers create a new class containing the shared attributes and behaviors forming a hierarchy. Modifications made at the highest level of the hierarchy apply to the subclasses.*		
APCS.MOD.3.A: *Designate private visibility of instance variables to encapsulate the attributes of an object.*		
	APCS.MOD.3.A.1	Data encapsulation is a technique in which the implementation details of a class are kept hidden from the user.
	APCS.MOD.3.A.2	When designing a class, programmers make decisions about what data to make accessible and modifiable from an external class. Data can be either accessible or modifiable, or it can be both or neither.
	APCS.MOD.3.A.3	Instance variables are encapsulated by using the private access modifier.
	APCS.MOD.3.A.4	The provided accessor and mutator methods in a class allow client code to use and modify data.
APCS.MOD.3.B: *Create an inheritance relationship from a subclass to the superclass.*		
	APCS.MOD.3.B.1	A class hierarchy can be developed by putting common attributes and behaviors of related classes into a single class called a superclass.
	APCS.MOD.3.B.2	Classes that extend a superclass, called subclasses, can draw upon the existing attributes and behaviors of the superclass without repeating these in the code.
	APCS.MOD.3.B.3	Extending a subclass from a superclass creates an "is-a" relationship from the subclass to the superclass.

	APCS.MOD.3.B.4	The keyword extends is used to establish an inheritance relationship between a subclass and a superclass. A class can extend only one superclass.
	APCS.MOD.3.B.5	Constructors are not inherited.
	APCS.MOD.3.B.6	The superclass constructor can be called from the first line of a subclass constructor by using the keyword super and passing appropriate parameters.
	APCS.MOD.3.B.7	The actual parameters passed in the call to the superclass constructor provide values that the constructor can use to initialize the object's instance variables.
	APCS.MOD.3.B.8	When a subclass's constructor does not explicitly call a superclass's constructor using super, Java inserts a call to the superclass's no-argument constructor.
	APCS.MOD.3.B.9	Regardless of whether the superclass constructor is called implicitly or explicitly, the process of calling superclass constructors continues until the Object constructor is called. At this point, all of the constructors within the hierarchy execute beginning with the Object constructor.
	APCS.MOD.3.B.10	Method overriding occurs when a public method in a subclass has the same method signature as a public method in the superclass.
	APCS.MOD.3.B.11	Any method that is called must be defined within its own class or its superclass.
	APCS.MOD.3.B.12	A subclass is usually designed to have modified (overridden) or additional methods or instance variables.
	APCS.MOD.3.B.13	A subclass will inherit all public methods from the superclass; these methods remain public in the subclass.
	APCS.MOD.3.B.14	The keyword super can be used to call a superclass's constructors and methods.
	APCS.MOD.3.B.15	The superclass method can be called in a subclass by using the keyword super with the method name and passing appropriate parameters.
	APCS.MOD.3.C: *Define reference variables of a superclass to be assigned to an object of a subclass in the same hierarchy.*	
	APCS.MOD.3.C.1	When a class S "is-a" class T, T is referred to as a superclass, and S is referred to as a subclass.
	APCS.MOD.3.D: *Call methods in an inheritance relationship.*	
	APCS.MOD.3.D.1	Utilize the Object class through inheritance.
	APCS.MOD.3.D.2	At compile time, methods in or inherited by the declared type determine the correctness of a non-static method call.
	APCS.MOD.3.D.3	At run-time, the method in the actual object type is executed for a non-static method call.

APCS.MOD.3.E: *Call object class methods through inheritance.*		
	APCS.MOD.3.E.1	The Object class is the superclass of all other classes in Java.
	APCS.MOD.3.E.2	The Object class is part of the java.langpackage
	APCS.MOD.3.E.3	The following Object class methods and constructors—including what they do and when they are used—are part of the Java Quick Reference: 1) boolean equals(Object other) 2) String toString(
	APCS.MOD.3.E.4	Subclasses of Object often override the equals and toString methods with class-specific implementations.
APCS.VAR.1: *Variables - To find specific solutions to generalizable problems, programmers include variables in their code so that the same algorithm runs using different input values.*		
	APCS.VAR.SK: *Suggested Skills*	
	APCS.VAR.SK.1.A	Determine an appropriate program design to solve a problem or accomplish a task (not assessed).
	APCS.VAR.SK.1.B	Determine code that would be used to complete code segments.
	APCS.VAR.SK.1.C	Determine code that would be used to complete code segments.
	APCS.VAR.SK.2.A	Apply the meaning of specific operators.
	APCS.VAR.SK.2.B	Determine the result or output based on statement execution order in a code segment without method calls (other than output).
	APCS.VAR.SK.2.C	Determine the result or output based on the statement execution order in a code segment containing methods calls.
	APCS.VAR.SK.2.E	Determine the number of times a code segment will execute.
	APCS.VAR.SK.3.A	Write program code to create objects of a class and call methods.
	APCS.VAR.SK.3.B	Write a program code to define a new type by creating a class.
	APCS.VAR.SK.3.D	Write program code to create, traverse, and manipulate elements in 1D array or ArrayList objects.
	APCS.VAR.SK.3.E	Write program code to create, traverse, and manipulate elements in 1D array or ArrayList objects.
	APCS.VAR.SK.4.A	Use test-cases to find errors or validate results.
	APCS.VAR.SK.4.B	Identify errors in program code.

	APCS.VAR.SK.4.C	Determine if two or more code segments yield equivalent results.
	APCS.VAR.SK.5.B	Explain why a code segment will not compile or work as intended.
APCS.VAR.1.A: *Create string literals.*		
	APCS.VAR.1.A.1	A string literal is enclosed in double quotes.
APCS.VAR.1.B: *Identify the most appropraite data type category for a particular specification.*		
	APCS.VAR.1.B.1	A type is a set of values (a domain) and a set of operations on them.
	APCS.VAR.1.B.2	Data types can be categorized as either primitive or reference.
	APCS.VAR.1.B.3	The primitive data types used in this course define the set of operations for numbers and Boolean values.
APCS.VAR.1.C: *Declare variables of the correct types to represent primitive data.*		
	APCS.VAR.1.C.1	The three primitive data types used in this course are int, double, and boolean.
	APCS.VAR.1.C.2	Each variable has associated memory that is used to hold its value.
	APCS.VAR.1.C.3	The memory associated with a variable of a primitive type holds an actual primitive value
	APCS.VAR.1.C.4	When a variable is declared final, its value cannot be changed once it is initialized.
APCS.VAR.1.D: *Define variables of the correct types to represent reference data.*		
	APCS.VAR.1.D	The keyword null is a special value used to indicate that a reference is not associated with any object.
APCS.VAR.1.E: *For string class: a.) Create string objects. b.) Call string methods*		
	APCS.VAR.1.E.1	String objects can be created by using string literals or by calling the String class constructor.
	APCS.VAR.1.E.2	String objects are immutable, meaning that String methods do not change the String object.
	APCS.VAR.1.E.3	String objects can be concatenated using the + or += operator, resulting in a new String object.
	APCS.VAR.1.E.4	Primitive values can be concatenated with a String object. This causes implicit conversion of the values to String objects.
	APCS.VAR.1.E.5	Escape sequences start with a \\ and have a special meaning in Java. Escape sequences used in this course include \\", \\\\, and \\n.
	APCS.VAR.1.E.6	Application program interfaces (APIs) and libraries simplify complex programming tasks.

	APCS.VAR.1.E.7	Documentation for APIs and libraries are essential to understanding the attributes and behaviors of an object of a class.
	APCS.VAR.1.E.8	Classes in the APIs and libraries are grouped into packages.
	APCS.VAR.1.E.9	The String class is part of the java.langpackage. Classes in the java.lang package are available by default.
	APCS.VAR.1.E.10	A String object has index values from 0 to length– 1. Attempting to access indices outside this range will result in an IndexOutOfBoundsException.
	APCS.VAR.1.E.11	A String object can be concatenated with an object reference, which implicitly calls the referenced object's toString method.
	APCS.VAR.1.E.12	The following String methods and constructors—including what they do and when they are used—are part of the Java Quick Reference:§String(String str) — Constructs a new String object that represents the same sequence of characters as str§int length() — Returns the number of characters in a String objectString substring(int from, int to) — Returns the substring beginning at index from and ending at index to − 1§§String substring(int from)— Returns substring(from, length())§int indexOf(String str) — Returns the index of the first occurrence of str; returns -1 if not found§boolean equals(String other)— Returns true if this is equal to other; returns false otherwise§int compareTo(String other)— Returns a value < 0 if this is less than other; returns zero if this is equal to other; returns a value > 0 if this is greater than other
	APCS.VAR.1.E.13	A string identical to the single element substring at position index can be created by calling substring(index, index + 1).
	APCS.VAR.1.F: *For wrapper classes: a.) Create integers objects. b.) Call integer methods. c.) Create double objects. d.) Call double methods.*	
	APCS.VAR.1.F.1	The Integer class and Double class are part of the java.lang package.
	APCS.VAR.1.F.2	The following Integer methods and constructors—including what they do and when they are used—are part of the Java Quick Reference:§Integer(int value) — Constructs a new Integer object that represents the specified int value§Integer.MIN_VALUE — The minimum value represented by an int or Integer§Integer.MAX_VALUE — The maximum value represented by an int or Integer§int intValue() — Returns the value of this Integer as an int
	APCS.VAR.1.F.3	The following Double methods and constructors—including what they do and when they are used—are part of the Java Quick Reference:Double(double value) —Constructs a new Double object that represents the specified double value§§double doubleValue() — Returns the value of this Double as a double

	APCS.VAR.1.F.4	Autoboxing is the automatic conversion that the Java compiler makes between primitive types and their corresponding object wrapper classes. This includes converting an int to an Integer and a double to a Double.
	APCS.VAR.1.F.5	The Java compiler applies autoboxing when a primitive value is:§Passed as a parameter to a method that expects an object of the corresponding wrapper class.§Assigned to a variable of the corresponding wrapper class.
	APCS.VAR.1.F.6	Unboxing is the automatic conversion that the Java compiler makes from the wrapper class to the primitive type. This includes converting an Integer to an int and a Double to a double.
	APCS.VAR.1.F.7	The Java compiler applies unboxing when a wrapper class object is:§Passed as a parameter to a method that expects a value of the corresponding primitive type.§Assigned to a variable of the corresponding primitive type.
	APCS.VAR.1.G: *Explain where variables can be used in the program code.*	
	APCS.VAR.1.G.1	Local variables can be declared in the body of constructors and methods. These variables may only be used within the constructor or method and cannot be declared to be publicor private.
	APCS.VAR.1.G.2	When there is a local variable with the same name as an instance variable, the variable name will refer to the local variable instead of the instance variable.
	APCS.VAR.1.G.3	Formal parameters and variables declared in a method or constructor can only be used within that method or constructor.
	APCS.VAR.1.G.4	Through method decomposition, a programmer breaks down a large problem into smaller subproblems by creating methods to solve each individual subproblem.
	APCS.VAR.1.H: *Evaluate object reference expressions that use the keyword this.*	
	APCS.VAR.1.H.1	Within a non-static method or a constructor, the keyword this is a reference to the current object—the object whose method or constructor is being called.
	APCS.VAR.1.H.2	The keyword this can be used to pass the current object as an actual parameter in a method call.
APCS.VAR.2: *Variables - To manage large amounts of data or complex relationships in data, programmers write code that groups the data together into a single data structure without creating individual variables for each value.*		
	APCS.VAR.2.A: *Represent collections of related primitive or object reference data using one-dimensional (1D) array objects.*	
	APCS.VAR.2.A.1	The use of array objects allows multiple related items to be represented using a single variable.
	APCS.VAR.2.A.2	The size of an array is established at the time of creation and cannot be changed.

	APCS.VAR.2.A.3	Arrays c.an store either primitive data or object reference data
	APCS.VAR.2.A.4	When an array is created using the keyword new, all of its elements are initialized with a specific value based on the type of elements: 1) Elements of type int are initialized to 0 2) Elements of type double are initialized to 0.0 3) Elements of type boolean are initialized to false 4) Elements of a reference type are initialized to the reference value null. No objects are automatically created
	APCS.VAR.2.A.5	Initializer lists can be used to create and initialize arrays.
	APCS.VAR.2.A.6	Square brackets ([]) are used to access and modify an element in a 1D array using an index.
	APCS.VAR.2.A.7	The .valid index values for an array are 0 through one less than the number of elements in the array, inclusive. Using an index value outside of this range will result in an ArrayIndexOutOfBoundsExceptionbeing thrown
APCS.VAR.2.B: *Traverse the elements in a 1D array.*		
	APCS.VAR.2.B.1	Iteration statements can be used to access all the elements in an array. This is called traversing the array.
APCS.VAR.2.C: *Traverse the elements in a 1D array object using an enhanced for loop.*		
	APCS.VAR.2.C.1	An enhanced for loop header includes a variable, referred to as the enhanced forloop variable.
	APCS.VAR.2.C.2	For each iteration of the enhanced for loop, the enhanced for loop variable is assigned a copy of an element without using its index.
	APCS.VAR.2.C.3	Assigning a new value to the enhanced forloop variable does not change the value stored in the array.
	APCS.VAR.2.C.4	Program code written using an enhanced forloop to traverse and access elements in an array can be rewritten using an indexed forloop or a while loop.
APCS.VAR.2.D: *Represent collections of related object reference data using arraylist objects*		
	APCS.VAR.2.D.1	An ArrayList object is mutable and contains object references.
	APCS.VAR.2.D.2	The ArrayList constructor ArrayList() constructs an empty list.
	APCS.VAR.2.D.3	Java allows the generic type ArrayList, where the generic type Especifies the type of the elements.
	APCS.VAR.2.D.4	When ArrayList is specified, the types of the reference parameters and return type when using the methods are type E.
	APCS.VAR.2.D.5	ArrrayList is preferred over ArrayList because it allows the compiler to find errors that would otherwise be found at run-time.

	APCS.VAR.2.D.6	The ArrayList class is part of the java.util package. An import statement can be used to make this class available for use in the program.
	APCS.VAR.2.D.7	The following ArrayList methods—including what they do and when they are used—are part of the Java Quick Reference: 1)int size() -Returns the number of elements in the list 2) boolean add(E obj) - Appendsobjto end of list; returnstrue 3) void add(int index, E obj) - Insertsobj at positionindex (0 <=index <= size),moving elements at positionindexand higher to the right (adds 1 to their indices) and adds 1 to sizeE 4) get(int index) - Returns the element at positionindexin the list 5) E set(int index, E obj) — Replaces the element at positionindex withobj;returns the element formerly at positionindex 6) E remove(int index) — Removes element from positionindex, moving elements at positionindex + 1and higher to the left (subtracts 1 from their indices) and subtracts 1 from size; returns the element formerly at positionindex
APCS.VAR.2.E: *For arraylist objects: Traverse using a for or while loop, Traverse using an enhanced for loop.*		
	APCS.VAR.2.E.1	Iteration statements can be used to access all the elements in an ArrayList. This is called traversing the ArrayList.
	APCS.VAR.2.E.2	Deleting elements during a traversal of an ArrayList requires using special techniques to avoid skipping elements.
	APCS.VAR.2.E.3	Since the indices for an ArrayListstart at 0 and end at the number of elements − 1, accessing an index value outside of this range will result in an ArrayIndexOutOfBoundsExceptionbeing thrown.
	APCS.VAR.2.E.4	Changing the size of an ArrayListwhile traversing it using an enhanced for loop can result in a ConcurrentModificationExceptionbeing thrown. Therefore, when using an enhanced for loop to traverse an ArrayList, you should not add or remove elements.
APCS.VAR.2.F: *Represent collections of related primitive or object reference data using two-dimensional (2D) array objects.*		
	APCS.VAR.2.F.1	2D arrays are stored as arrays of arrays. Therefore, the way 2D arrays are created and indexed is similar to 1D array objects.
	APCS.VAR.2.F.2	For the purposes of the exam, when accessing the element at arr[first][second], the first index is used for rows, the second index is used for columns.
	APCS.VAR.2.F.3	The initializer list used to create and initialize a 2D array consists of initializer lists that represent 1D arrays.
	APCS.VAR.2.F.4	The square brackets [row][col] are used to access and modify an element in a 2D array.

	APCS.VAR.2.F.5	"Row-major order" refers to an ordering of 2D array elements where traversal occurs across each row, while "column-major order" traversal occurs down each column.
APCS.VAR.2.G: *For 2D array objects: traverse using nested loops, traverse using nested enhanced for loops.*		
	APCS.VAR.2.G.1	Nested iteration statements are used to traverse and access all elements in a 2D array. Since 2D arrays are stored as arrays of arrays, the way 2D arrays are traversed using forloops and enhanced for loops is similar to 1D array objects.
	APCS.VAR.2.G.2	Nested iteration statements can be written to traverse the 2D array in "row-major order" or "column-major order."
	APCS.VAR.2.G.3	The outer loop of a nested enhanced forloop used to traverse a 2D array traverses the rows. Therefore, the enhanced for loop variable must be the type of each row, which is a 1D array. The inner loop traverses a single row. Therefore, the inner enhanced forloop variable must be the same type as the elements stored in the 1D array.
APCS.CON.1: *Control - The way variables and operators are sequenced and combined in an expression determines the computed result.*		
APCS.CON.SK: *Suggested Skills*		
	APCS.CON.SK.1.A	Determine an appropriate program design to solve a problem or accomplish a task (not assessed).
	APCS.CON.SK.1.B	Determine code that would be used to complete code segments.
	APCS.CON.SK.1.C	Determine code that would be used to complete code segments.
	APCS.CON.SK.2.A	Apply the meaning of specific operators.
	APCS.CON.SK.2.B	Determine the result or output based on statement execution order in a code segment without method calls (other than output).
	APCS.CON.SK.2.C	Determine the result or output based on statement execution order in a code segment without method calls (other than output).
	APCS.CON.SK.2.D	Determine the number of times a code segment will execute.
	APCS.CON.SK.2.E	Determine the number of times a code segment will execute.
	APCS.CON.SK.3.A	Write program code to create objects of a class and call methods.
	APCS.CON.SK.3.B	Write program code to define a new type by creating a class.
	APCS.CON.SK.3.C	Write program code to create objects of a class and call methods.

	APCS.CON.SK.3.D	Write program code to create, traverse, and manipulate elements in 1D array or ArrayList objects.
	APCS.CON.SK.3.E	Write program code to create, traverse, and manipulate elements in 2D array objects.
	APCS.CON.SK.4.A	Use test-cases to find error or validate results.
	APCS.CON.SK.4.B	Identify errors in program code.
	APCS.CON.SK.4.C	Determine if two or more code segments yield equivalent results.
	APCS.CON.SK.5.A	Describe the behavior of a given segment of program code.
	APCS.CON.SK.5.B	Explain why a code segment will not compile or work as intended.
	APCS.CON.SK.5.C	Explain how the result of program code chages, given a change to the initial code.
	APCS.CON.SK.5.D	Describe the initial conditions that must be met for a program segment to work as intended or described.
APCS.CON.1.A: *Evaluate arthimetic expressions in a program code.*		
	APCS.CON.1.A.1	A literal is the source code representation of a fixed value.
	APCS.CON.1.A.2	Arithmetic expressions include expressions of type int and double.
	APCS.CON.1.A.3	The arithmetic operators consist of +, −, *, /, and %.
	APCS.CON.1.A.4	An arithmetic operation that uses two intvalues will evaluate to an int value.
	APCS.CON.1.A.5	An arithmetic operation that uses a doublevalue will evaluate to a double value.
	APCS.CON.1.A.6	Operators can be used to construct compound expressions.
	APCS.CON.1.A.7	During evaluation, operands are associated with operators according to operator precedence to determine how they are grouped.
	APCS.CON.1.A.8	An attempt to divide an integer by zero will result in an ArithmeticException to occur.
APCS.CON.1.B: *Evaluate what is stored in a variable as a result of an expression with an assignment statement.*		
	APCS.CON.1.B.1	The assignment operator (=) allows a program to initialize or change the value stored in a variable. The value of the expression on the right is stored in the variable on the left.
	APCS.CON.1.B.2	During execution, expressions are evaluated to produce a single value.

	APCS.CON.1.B.3	The value of an expression has a type based on the evaluation of the expression.
	APCS.CON.1.B.4	Compound assignment operators (+=, −=, *=, /=, %=) can be used in place of the. assignment operator
	APCS.CON.1.B.5	The increment operator (++) and decrement operator (−−) are used to add 1 or subtract 1 from the stored value of a variable or an array element. The new value is assigned to the variable or array element.
APCS.CON.1.C: *Evaluate arithmetic expressions that use casting.*		
	APCS.CON.1.C.1	The casting operators (int) and (double)can be used to create a temporary value converted to a different data type
	APCS.CON.1.C.2	Casting a double value to an int causes the digits to the right of the decimal point to be truncated.
	APCS.CON.1.C.3	Some programming code causes int values to be automatically cast (widened) to doublevalues.
	APCS.CON.1.C.4	Values of type double can be rounded to the nearest integer by (int)(x + 0.5) or (int)(x - 0.5) for negative numbers.
	APCS.CON.1.C.5	Integer values in Java are represented by values of type int, which are stored using a finite amount (4 bytes) of memory. Therefore, an int value must be in the range from Integer values in Java are represented by values of type int, which are stored using a finite amount (4 bytes) of memory. Therefore, an int value must be in the range from Integer.MIN_VALUE to Integer.MAX_VALUE inclusive.
	APCS.CON.1.C.6	If an expression would evaluate to an intvalue outside of the allowed range, an integer overflow occurs. This could result in an incorrect value within the allowed range.
APCS.CON.1.D: *Evaluate expressions that use the Math class methods.*		
	APCS.CON.1.D.1	The Math class is part of the java.langpackage.
	APCS.CON.1.D.2	The Math class contains only static methods
	APCS.CON.1.D.3	The following static Math methods—including what they do and when they are used—are part of the Java Quick Reference:§int abs(int x) — Returns the absolute value of an int value§double abs(double x) — Returns the absolute value of a double value§double pow(double base, double exponent) — Returns the value of the first parameter raised to the power of the second parameter§double sqrt(double x) — Returns the positive square root of a double value§double random() — Returns a double value greater than or equal to 0.0 and less than 1.0

	APCS.CON.1.D.4	The values returned from Math.random can be manipulated to produce a random int or double in a defined range.
APCS.CON.1.E: *Evaluate Boolean expressions that use relational operators in program code.*		
	APCS.CON.1.E.1	Primitive values and reference values can be compared using relational operators (i.e., == and !=).
	APCS.CON.1.E.2	Arithmetic expression values can be compared using relational operators (i.e., <, >, <=, >=).
	APCS.CON.1.E.3	An expression involving relational operators evaluates to a Boolean value.
APCS.CON.1.F: *Evaluate compound Boolean expressions in program code.*		
	APCS.CON.1.F.1	Logical operators !(not), &&(and), and ‖(or) are used with Boolean values. This represents the order these operators will be evaluated.
	APCS.CON.1.F.2	An expression involving logical operators evaluates to a Boolean value.
	APCS.CON.1.F.3	When the result of a logical expression using && or ‖ can be determined by evaluating only the first Boolean operand, the second is not evaluated. This is known as short-circuited evaluation.
APCS.CON.1.G: *Compare and contrast equivalent Boolean expressions.*		
	APCS.CON.1.G.1	De Morgan's Laws can be applied to Boolean expressions.
	APCS.CON.1.G.2	Truth tables can be used to prove Boolean identities.
	APCS.CON.1.G.3	Equivalent Boolean expressions will evaluate to the same value in all cases.
APCS.CON.1.H: *Compare object references using Boolean expressions in program code.*		
	APCS.CON.1.H.1	Two object references are considered aliases when they both reference the same object.
	APCS.CON.1.H.2	Object reference values can be compared, using == and !=, to identify aliases.
	APCS.CON.1.H.3	A reference value can be compared with null, using == or !=, to determine if the reference actually references an object.
	APCS.CON.1.H.4	Often classes have their own equals method, which can be used to determine whether two objects of the class are equivalent.
APCS.CON.2: *Control - rogrammers incorporate iteration and selection into code as a way of providing instructions for the computer to process each of the many possible input values.*		
APCS.CON.2.A: *Represent branching logical processes by using conditional statements.*		
	APCS.CON.2.A.1	Conditional statements interrupt the sequential execution of statements.

	APCS.CON.2.A.2	if statements affect the flow of control by executing different statements based on the value of a Boolean expression.
	APCS.CON.2.A.3	A one-way selection (if statement) is written when there is a set of statements to execute under a certain condition. In this case, the body is executed only when the Boolean condition is true.
	APCS.CON.2.A.4	A two-way selection is written when there are two sets of statements— one to be executed when the Boolean condition is true, and another set for when the Boolean condition is false. In this case, the body of the "if" is executed when the Boolean condition is true, and the body of the "else" is executed when the Boolean condition is false.
	APCS.CON.2.A.5	A multi-way selection is written when there are a series of conditions with different statements for each condition. Multi-way selection is performed using if-else-if statements such that exactly one section of code is executed based on the first condition that evaluates to true.
APCS.CON.2.B: *Represent branching logical processes by using nested conditional statements.*		
	APCS.CON.2.B.1	Nested if statements consist of ifstatements within if statements.
APCS.CON.2.C: *Represent iterative process using a while loop.*		
	APCS.CON.2.C.1	Iteration statements change the flow of control by repeating a set of statements zero or more times until a condition is met.
	APCS.CON.2.C.2	In loops, the Boolean expression is evaluated before each iteration of the loop body, including the first. When the expression evaluates to true, the loop body is executed. This continues until the expression evaluates to false, whereupon the iteration ceases.
	APCS.CON.2.C.3	A loop is an infinite loop when the Boolean expression always evaluates to true.
	APCS.CON.2.C.4	If the Boolean expression evaluates to falseinitially, the loop body is not executed at all.
	APCS.CON.2.C.5	Executing a return statement inside an iteration statement will halt the loop and exit the method or constructor.
APCS.CON.2.D: *For algorithms in the context of a particular specification that does not require the use of transversals: identify standard algorithms, modify standard algorithms, develop an algorithm.*		
	APCS.CON.2.D.1	There are standard algorithms to: 1) Identify if an integer is or is not evenly divisible by another integer 2) Identify the individual digits in an integer 3)Determine the frequency with which a specific criterion is met.
	APCS.CON.2.D.2	There are standard algorithms to: 1)Determine a minimum or maximum value 2) Compute a sum, average, or mode
APCS.CON.2.E: *Represent iterative processes using a for loop.*		

	APCS.CON.2.E.1	There are three parts in a for loop header: the initialization, the Boolean expression, and the increment. The increment statement can also be a decrement statement.
	APCS.CON.2.E.2	In a for loop, the initialization statement is only executed once before the first Boolean expression evaluation. The variable being initialized is referred to as a loop control variable.
	APCS.CON.2.E.3	In each iteration of a for loop, the increment statement is executed after the entire loop body is executed and before the Boolean expression is evaluated again.
	APCS.CON.2.E.4	A for loop can be rewritten into an equivalent while loop and vice versa.
	APCS.CON.2.E.5	"Off by one" errors occur when the iteration statement loops one time too many or one time too few.
	APCS.CON.2.F: *For algorithms in the context of a particular specification that involves string objects: identify standard algorithms, modify standard algorithms, and develop an algorithm.*	
	APCS.CON.2.F.1	There are standard algorithms that utilize String traversals to: 1) Find if one or more substrings has a particular property 2) Determine the number of substrings that meet specific criteria 3) Create a new string with the characters reversed
	APCS.CON.2.G: *Represent nested iterative processes.*	
	APCS.CON.2.G.1	Nested iteration statements are iteration statements that appear in the body of another iteration statement.
	APCS.CON.2.G.2	When a loop is nested inside another loop, the inner loop must complete all its iterations before the outer loop can continue.
	APCS.CON.2.H: *Computer statement execution counts and informal run-time comparison of iterative statements.*	
	APCS.CON.2.H.1	A statement execution count indicates the number of times a statement is executed by the program.
	APCS.CON.2.I: *For algorithms in the context of a particular specification that requires the use of array traversals: identify standard algorithms, modify standard algorithms, develop an algorithm.*	
	APCS.CON.2.I.1	There are standard algorithms that utilize array traversals to: 1) Determine a minimum or maximum value 2) Compute a sum, average, or mode 3) Determine if at least one element has a particular property 4) Determine if all elements have a particular property 5) Access all consecutive pairs of elements 6) Determine the presence or absence of duplicate elements 7) Determine the number of elements meeting specific criteria
	APCS.CON.2.I.2	There are standard array algorithms that utilize traversals to: 1) Shift or rotate elements left or right 2) Reverse the order of the elements

	APCS.CON.2.J: *For algorithms in the context of a particular specification that requires the use of arraylist traversals: identify standard algorithms, modify standard algorithms, develop an algorithm.*	
	APCS.CON.2.J.1	There are standard ArrayList algorithms that utilize traversals to: 1) Insert elements 2) Delete elements 3) Apply the same standard algorithms that are used with 1D array
	APCS.CON.2.J.2	Some algorithms require multiple String, array, or ArrayList objects to be traversed simultaneously.
	APCS.CON.2.K: *Apply sequential/linear search algorithms to search for specific information in array or arraylist objects.*	
	APCS.CON.2.K.1	There are standard algorithms for searching.
	APCS.CON.2.K.2	Sequential/linear search algorithms check each element in order until the desired value is found or all elements in the array or ArrayList have been checked.
	APCS.CON.2.L: *Apply selection sort and insertion algorithms to sort the elements of array or arraylist objects.*	
	APCS.CON.2.L	Selection sort and insertion sort are iterative sorting algorithms that can be used to sort elements in an array or ArrayList.
	APCS.CON.2.M: *Compute statement execution and informal run-time comparison of sorting algorithms.*	
	APCS.CON.2.M.1	Informal run-time comparisons of program code segments can be made using statement execution counts.
	APCS.CON.2.N: *For algorithms in the context of a particular specification that requires the use of 2D array traversals: identify standard algorithms, modify standard algorithms, develop an algorithm.*	
	APCS.CON.2.N.1	When applying sequential/linear search algorithms to 2D arrays, each row must be accessed then sequential/linear search applied to each row of a 2D array.
	APCS.CON.2.N.2	All standard 1.D array algorithms can be applied to 2D array objects
	APCS.CON.2.O: *Determine the result of executing recursive methods.*	
	APCS.CON.2.O.1	A recursive method is a method that calls itself.
	APCS.CON.2.O.2	Recursive methods contain at least one base case, which halts the recursion, and at least one recursive call.
	APCS.CON.2.O.3	Each recursive call has its own set of local variables, including the formal parameters.
	APCS.CON.2.O.4	Parameter values capture the progress of a recursive process, much like loop control variable values capture the progress of a loop.
	APCS.CON.2.O.5	Any recursive solution can be replicated through the use of an iterative approach.

	APCS.CON.2.O.6	Recursion can be used to traverse String, array, and ArrayList objects.
APCS.CON.2.P: *Apply recursive search algorithms to information in string, 1D array, or arraylist objects.*		
	APCS.CON.2.P.1	Data must be in sorted order to use the binary search algorithm.
	APCS.CON.2.P.2	The binary search algorithm starts at the middle of a sorted array or ArrayList and eliminates half of the array or ArrayList in each iteration until the desired value is found or all elements have been eliminated.
	APCS.CON.2.P.3	Binary search can be more efficient than sequential/linear search.
	APCS.CON.2.P.4	The binary search algorithm can be written either iteratively or recursively.
APCS.CON.2.Q: *Apply recursive algorithms to sort elements of array or arraylist objects.*		
	APCS.CON.2.Q.1	Merge sort is a recursive sorting algorithm that can be used to sort elements in an array or ArrayList.
APCS.IOC.1: *Impact of Computing - While programs are typically designed to achieve a specific purpose, they may have unintended consequences.*		
APCS.IOC.1.A: *Explain the ethical and social implications of computing systems.*		
	APCS.IOC.1.A.1	System reliability is limited. Programmers should make an effort to maximize system reliability.
	APCS.IOC.1.A.2	Legal issues and intellectual property concerns arise when creating programs.
	APCS.IOC.1.A.3	The creation of programs has impacts on society, economies, and culture. These impacts can be beneficial and/or harmful.
APCS.IOC.1.B: *Explain the risks to privacy from collecting and storing personal data on computer systems.*		
	APCS.IOC.1.B.1	When using the computer, personal privacy is at risk. Programmers should attempt to safeguard personal privacy.
	APCS.IOC.1.B.2	Computer use and the creation of programs have an impact on personal security. These impacts can be beneficial and/or harmful.