Trinity Area School District
Template for Curriculum Mapping

Course: Computer Science I **Grade: 9 - 12**	**Overview of Course:** As the world turns to graphical user interfaces (aka Windows), computer-programming languages are changing to accommodate the shift. This course will explore programming languages such as: Scratch; Alice; Python; and App Inventor. These languages are designed to allow the programmer to develop complex applications without the complexity generally associated with Windows programming. Scratch, Alice, Python, and App Inventor designed in a way that is more intuitive for students, which makes them excellent tools for understanding elementary programming concepts. The students will use Scratch, Alice, Python, and App Inventor to create object-oriented, event-driven programs. Objects/controls are covered including their properties, methods, and events. Additional topics include hardware, networking, developing algorithms, binary, octal, and hexadecimal number systems, proper programming conventions, variables, calculations, conditional statements, iteration, functions, and various programming techniques as they apply to Scratch, Alice, Python, and App Inventor. Logic and problem-solving skills are developed and implemented throughout the course.

Overarching Big Ideas, Enduring Understandings, and Essential Questions

Big Idea	Standard(s) Addressed **Standards Used:** **CSTA K – 12 Computer Science Standards** http://csta.acm.org/Curriculum/sub/K12Standards.html	Enduring Understanding(s)	Essential Question(s)

Algorithm Development	**5.3.A Computer Science in the Modern World - Computational Thinking** 1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. 2. Describe a software development process used to solve software problems (e.g., design, coding, testing, verification). 3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. 5. Describe the relationship between binary and hexadecimal representations. 7. Describe how various types of data are stored in a computer system. 8. Use modeling and simulation to represent and understand natural phenomena. 10. Describe the concept of parallel processing as a strategy to solve	Algorithmic thinking is a crucial process that students learn to facilitate problem solving. Breaking a problem down into smaller steps, analyzing possible solutions, and testing results are necessary tasks in solving even the simplest problem.	How will developing an algorithm to solve a problem be better than trying "quick fix" solutions until the solution is discovered?

	large problems. 11. Describe how computation shares features with art and music by translating human intention into an artifact. **5.3.A Computer Science in the Modern World - Collaboration** 3. Describe how computing enhances traditional forms and enables new forms of experience, expression, communication, and collaboration. **5.3.A Computer Science in the Modern World - Computing Practice and Programming** 2. Use mobile devices/emulators to design, develop, and implement mobile computing applications. 3. Use various debugging and testing methods to ensure program correctness (e.g., test cases, unit testing, white box, black box, integration testing). 4. Apply analysis, design, and implementation techniques to solve problems (e.g., use one or more software lifecycle models). 5. Use Application Program Interfaces (APIs) and libraries to facilitate programming solutions. 6. Select appropriate file formats for various types and uses of data. 7. Describe a variety of programming languages available to solve problems and develop systems. 8. Explain the program execution process. 10. Explore a variety of careers to which computing is central. 12. Describe how mathematical and statistical functions, sets, and logic are used in computation. **5.3.A Computer Science in the Modern World - Computers and Communications Devices** 1. Describe the unique features of computers embedded in mobile devices and vehicles (e.g., cell phones, automobiles, airplanes). 2. Develop criteria for purchasing or upgrading computer system hardware.		

	3. Describe the principal components of computer organization (e.g., input, output, processing, and storage).		

	4. Compare various forms of input and output. 5. Explain the multiple levels of hardware and software that support program execution (e.g., compilers, interpreters, operating systems, networks). **5.3.A Computer Science in the Modern World - Community, Global, and Ethical Impacts** 1. Compare appropriate and inappropriate social networking behaviors. 9. Describe different ways in which software is created and shared and their benefits and drawbacks (commercial software, public domain software, open source development). **5.3.B Computer Science Concepts and Practices - Computational Thinking** 4. Evaluate algorithms by their efficiency, correctness, and clarity. 7. Discuss the interpretation of binary sequences in a variety of forms (e.g., instructions, numbers, text, sound, and image). 8. Use models and simulations to help formulate, refine, and test scientific hypotheses. 10. Decompose a problem by defining new functions and classes. 11. Demonstrate concurrency by separating processes into threads and dividing data into parallel streams. **5.3.B Computer Science Concepts and Practices - Computing Practice and Programming** 1. Use advanced tools to create digital artifacts (e.g., web design, animation, video, multimedia). 2. Use tools of abstraction to decompose a large-scale computational problem (e.g., procedural abstraction, objectoriented design, functional design). 3. Classify programming languages based on their level and application domain 6. Anticipate future careers and the technologies that will exist.		

	5.3.B Computer Science Concepts and Practices - Computers		

	and Communications Devices 1. Discuss the impact of modifications on the functionality of application programs. **5.3.B Computer Science Concepts and Practices - Community, Global, and Ethical Impacts** 2. Analyze the beneficial and harmful effects of computing innovations. 3. Summarize how financial markets, transactions, and predictions have been transformed by automation. 4. Summarize how computation has revolutionized the way people build real and virtual organizations and infrastructures. 7. Differentiate among open source, freeware, and proprietary software licenses and their applicability to different types of software. **5.3.C.2 Projects-Based Courses**		

Programming Conventions and Syntax	**5.3.A Computer Science in the Modern World - Computational Thinking** 1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. 2. Describe a software development process used to solve software problems (e.g., design, coding, testing, verification). 3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. 5. Describe the relationship between binary and hexadecimal representations. 7. Describe how various types of data are stored in a computer system. 8. Use modeling and simulation to represent and understand natural phenomena. 10. Describe the concept of parallel processing as a strategy to solve large problems. 11. Describe how computation shares features with art and music by translating human intention into an artifact.	Programming an exact solution to simple problems, building up to more complex problems is a model that can be transferred to solving problems that do not involve computers.	Why are programming conventions important? Why should we use a prescribed style for programming? If I learn one programming language, will that help me learn another?

	5.3.A Computer Science in the Modern World - Collaboration 3. Describe how computing enhances traditional forms and enables new forms of experience, expression, communication, and collaboration. **5.3.A Computer Science in the Modern World - Computing Practice and Programming** 2. Use mobile devices/emulators to design, develop, and implement mobile computing applications. 3. Use various debugging and testing methods to ensure program correctness (e.g., test cases, unit testing, white box, black box, integration testing). 4. Apply analysis, design, and implementation techniques to solve problems (e.g., use one or more software lifecycle models). 5. Use Application Program Interfaces (APIs) and libraries to facilitate programming solutions. 6. Select appropriate file formats for various types and uses of data. 7. Describe a variety of programming languages available to solve problems and develop systems. 8. Explain the program execution process. 10. Explore a variety of careers to which computing is central. 12. Describe how mathematical and statistical functions, sets, and logic are used in computation. **5.3.A Computer Science in the Modern World - Computers and Communications Devices** 1. Describe the unique features of computers embedded in mobile devices and vehicles (e.g., cell phones, automobiles, airplanes). 2. Develop criteria for purchasing or upgrading computer system hardware. 3. Describe the principal components of computer organization (e.g., input, output, processing, and storage). 4. Compare various forms of input and output.		

	5. Explain the multiple levels of hardware and software that support program execution (e.g., compilers, interpreters, operating systems,		

	networks). **5.3.A Computer Science in the Modern World - Community, Global, and Ethical Impacts** 1. Compare appropriate and inappropriate social networking behaviors. 9. Describe different ways in which software is created and shared and their benefits and drawbacks (commercial software, public domain software, open source development). **5.3.B Computer Science Concepts and Practices - Computational Thinking** 4. Evaluate algorithms by their efficiency, correctness, and clarity. 7. Discuss the interpretation of binary sequences in a variety of forms (e.g., instructions, numbers, text, sound, and image). 8. Use models and simulations to help formulate, refine, and test scientific hypotheses. 10. Decompose a problem by defining new functions and classes. 11. Demonstrate concurrency by separating processes into threads and dividing data into parallel streams. **5.3.B Computer Science Concepts and Practices - Computing Practice and Programming** 1. Use advanced tools to create digital artifacts (e.g., web design, animation, video, multimedia). 2. Use tools of abstraction to decompose a large-scale computational problem (e.g., procedural abstraction, objectoriented design, functional design). 3. Classify programming languages based on their level and application domain 6. Anticipate future careers and the technologies that will exist. **5.3.B Computer Science Concepts and Practices - Computers and Communications Devices**		

	1. Discuss the impact of modifications on the functionality of application programs.		

	5.3.B Computer Science Concepts and Practices - Community, Global, and Ethical Impacts 2. Analyze the beneficial and harmful effects of computing innovations. 3. Summarize how financial markets, transactions, and predictions have been transformed by automation. 4. Summarize how computation has revolutionized the way people build real and virtual organizations and infrastructures. 7. Differentiate among open source, freeware, and proprietary software licenses and their applicability to different types of software. **5.3.C.2 Projects-Based Courses**		
Collaborative Problem Solving	**5.3.A Computer Science in the Modern World - Collaboration** 1. Work in a team to design and develop a software artifact. 4. Identify how collaboration influences the design and development of software products. **5.3.B Computer Science Concepts and Practices - Collaboration** 1. Use project collaboration tools, version control systems, and Integrated Development Environments (IDEs) while working on a collaborative software project. 3. Evaluate programs written by others for readability and usability. **5.3.C.2 Projects-Based Courses**	Working with other students to develop solutions to problems will promote new ideas and facilitate discussion of ideas with peers.	Why is it necessary to collaborate when solving problems? What are the advantages and disadvantages of collaborative problem solving?
Big Ideas, Enduring Understandings, and Essential Questions Per Unit of Study			

Month of Instruction	Title of Unit	Big Idea(s)	Standard(s) Addressed **Standards Used:** **CSTA K – 12 Computer Science Standards** http://csta.acm.org/Curriculum/sub/K12Standards.html	Enduring Understanding(s)	Essential Question(s)	Common Assessment(s)*	Common Resource(s)* Used

August	**Alice Introduction**	**Introduction to Using the Alice 3D IDE**	**5.3.A Computer Science in the Modern World - Computational Thinking** 1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. 2. Describe a software development process used to solve software problems (e.g., design, coding, testing, verification). 3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. 7. Describe how various types of data are stored in a computer system. 8. Use modeling and simulation to represent and understand natural phenomena. 10. Describe the concept of parallel processing as a strategy to solve large problems. 11. Describe how computation shares features with art and music by translating human intention into an artifact. **5.3.A Computer Science in the Modern World - Collaboration** 1. Work in a team to design and develop a software artifact. 3. Describe how computing enhances traditional forms and enables new forms of experience, expression, communication, and collaboration. 4. Identify how collaboration influences the design and development of software products.	**Use in-class practice and exploration to increase knowledge of Alice programming conventions and syntax.** **Use specific Alice concepts to develop basic programming conventions and syntax.**	**What are the components of a 3D language that will facilitate solving a problem?** **What type of problems can be solved using the specific 3D programming components?**	**Participation** **Class Notes** **Quiz** **Presentation** **<u>Terms</u>** **Class** **Computer program** **Design** **Flowchart** **Object**	**On-Line Resources** **Teacherdeveloped notes** **Textbook**

			5.3.A Computer Science in the Modern World - **Computing Practice and Programming** 3. Use various debugging and testing methods to ensure program correctness (e.g., test cases, unit testing, white box, black box, integration testing). 4. Apply analysis, design, and implementation				

			techniques to solve problems (e.g., use one or more software lifecycle models). 5. Use Application Program Interfaces (APIs) and libraries to facilitate programming solutions. 6. Select appropriate file formats for various types and uses of data. 7. Describe a variety of programming languages available to solve problems and develop systems. 8. Explain the program execution process. 12. Describe how mathematical and statistical functions, sets, and logic are used in computation. **5.3.B Computer Science Concepts and Practices - Computational Thinking** 4. Evaluate algorithms by their efficiency, correctness, and clarity. 8. Use models and simulations to help formulate, refine, and test scientific hypotheses. 10. Decompose a problem by defining new functions and classes. 11. Demonstrate concurrency by separating processes into threads and dividing data into parallel streams. **5.3.B Computer Science Concepts and Practices - Collaboration** 1. Use project collaboration tools, version control systems, and Integrated Development Environments (IDEs) while working on a collaborative software project. 3. Evaluate programs written by others for readability and usability. **5.3.B Computer Science Concepts and Practices**				

			- Computing Practice and Programming 1. Use advanced tools to create digital artifacts				

			(e.g., web design, animation, video, multimedia). 2. Use tools of abstraction to decompose a largescale computational problem (e.g., procedural abstraction, object-oriented design, functional design). 3. Classify programming languages based on their level and application domain **5.3.C.2 Projects-Based Courses**				

September	**Alice Program Design and Implementation**	**Movement of Alice objects**	**5.3.A Computer Science in the Modern World - Computational Thinking** 1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. 2. Describe a software development process used to solve software problems (e.g., design, coding, testing, verification). 3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. 7. Describe how various types of data are stored in a computer system. 8. Use modeling and simulation to represent and understand natural phenomena. 10. Describe the concept of parallel processing as a strategy to solve large problems. 11. Describe how computation shares features with art and music by translating human intention into an artifact. **5.3.A Computer Science in the Modern World - Collaboration** 1. Work in a team to design and develop a software artifact. 3. Describe how computing enhances traditional forms and enables new forms of experience,	**Use Alice to move, resize, and manipulate objects.** **Use Alice to plan the interaction between objects.**	**Can the objects interact with each other?** **What are the limitations to the movements of Alice objects?** **Do all characters respond to movements the same way?**	**Participation** **Class Notes** **Quiz** **Presentation** **<u>Terms</u>** **Algorithm** **Argument** **Bug** **Comment** **Control structure** **Design** **Implement** **Instruction** **Method** **Nesting** **Property** **Pseudocode** **Runtime** **Scenario** **State** **Storyboard** **Syntax** **Trial-and-error**	**On-Line Resources** **Teacherdeveloped notes** **Textbook**

			expression, communication, and collaboration. 4. Identify how collaboration influences the design and development of software products. **5.3.A Computer Science in the Modern World - Computing Practice and Programming** 3. Use various debugging and testing methods to ensure program correctness (e.g., test cases, unit testing, white box, black box, integration testing). 4. Apply analysis, design, and implementation techniques to solve problems (e.g., use one or more software lifecycle models). 5. Use Application Program Interfaces (APIs) and libraries to facilitate programming solutions. 6. Select appropriate file formats for various types and uses of data. 7. Describe a variety of programming languages available to solve problems and develop systems. 8. Explain the program execution process. 12. Describe how mathematical and statistical functions, sets, and logic are used in computation. **5.3.B Computer Science Concepts and Practices - Computational Thinking** 4. Evaluate algorithms by their efficiency, correctness, and clarity. 8. Use models and simulations to help formulate, refine, and test scientific hypotheses. 10. Decompose a problem by defining new functions and classes. 11. Demonstrate concurrency by separating processes into threads and dividing data into parallel streams.				

			5.3.B Computer Science Concepts and Practices - Collaboration				

			1. Use project collaboration tools, version control systems, and Integrated Development Environments (IDEs) while working on a collaborative software project. 3. Evaluate programs written by others for readability and usability. **5.3.B Computer Science Concepts and Practices** **- Computing Practice and Programming** 1. Use advanced tools to create digital artifacts (e.g., web design, animation, video, multimedia). 2. Use tools of abstraction to decompose a largescale computational problem (e.g., procedural abstraction, object-oriented design, functional design). 3. Classify programming languages based on their level and application domain **5.3.C.2 Projects-Based Courses**				

September	**Alice Functions and Interactions**	**Functions and Expressions**	**5.3.A Computer Science in the Modern World - Computational Thinking** 1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. 2. Describe a software development process used to solve software problems (e.g., design, coding, testing, verification). 3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. 7. Describe how various types of data are stored in a computer system. 8. Use modeling and simulation to represent and understand natural phenomena. 10. Describe the concept of parallel processing as a strategy to solve large problems.	**Use Alice functions to determine movement.** **User math expressions to determine distance from object to object.** **Use Alice if/else with functions and math expressions to produce**	**How do functions and math expressions determine an object's movement and position?** **How do you determine a boolean value?** **How do boolean**	**Participation** **Class Notes** **Quiz** **Presentation** **<u>Terms</u>** **Boolean value** **Condition** **Conditional execution** **Control structure** **Count** **Decision** **Expression** **Function**	**On-Line Resources** **Teacherdeveloped notes** **Textbook**

			11. Describe how computation shares features with art and music by translating human intention into an artifact. **5.3.A Computer Science in the Modern World - Collaboration** 1. Work in a team to design and develop a software artifact. 3. Describe how computing enhances traditional forms and enables new forms of experience, expression, communication, and collaboration. 4. Identify how collaboration influences the design and development of software products. **5.3.A Computer Science in the Modern World - Computing Practice and Programming** 3. Use various debugging and testing methods to ensure program correctness (e.g., test cases, unit testing, white box, black box, integration testing). 4. Apply analysis, design, and implementation techniques to solve problems (e.g., use one or more software lifecycle models). 5. Use Application Program Interfaces (APIs) and libraries to facilitate programming solutions. 6. Select appropriate file formats for various types and uses of data. 7. Describe a variety of programming languages available to solve problems and develop systems. 8. Explain the program execution process. 12. Describe how mathematical and statistical functions, sets, and logic are used in computation. **5.3.B Computer Science Concepts and Practices**	**boolean results for conditional movement.** **Use Alice loop constructs to repeat movements.** **Explore and change points of view using Alice camera angles and positioning.**	**values affect if/else and loop execution?** **What real world situations require if/else?** **What real world situations require a loop?** **How does the camera angle affect the realism of the Alice animation?**	**Instruction** **Loop** **Relational operator** **Repetition**	

			- Computational Thinking 4. Evaluate algorithms by their efficiency, correctness, and clarity.				

			8. Use models and simulations to help formulate, refine, and test scientific hypotheses. 10. Decompose a problem by defining new functions and classes. 11. Demonstrate concurrency by separating processes into threads and dividing data into parallel streams. **5.3.B Computer Science Concepts and Practices** **- Collaboration** 1. Use project collaboration tools, version control systems, and Integrated Development Environments (IDEs) while working on a collaborative software project. 3. Evaluate programs written by others for readability and usability. **5.3.B Computer Science Concepts and Practices** **- Computing Practice and Programming** 1. Use advanced tools to create digital artifacts (e.g., web design, animation, video, multimedia). 2. Use tools of abstraction to decompose a largescale computational problem (e.g., procedural abstraction, object-oriented design, functional design). 3. Classify programming languages based on their level and application domain **5.3.C.2 Projects-Based Courses**				

October	**Classes, Object, Methods, and Parameters**	**Classes, Object, Methods, and Parameters**	**5.3.A Computer Science in the Modern World - Computational Thinking** 1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. 2. Describe a software development process used	**How classes, objects, methods, and parameters interrelate through**	**How does the structure of the class and its methods determine the parameter**	**Participation** **Class Notes** **Quiz** **Presentation** **<u>Terms</u>**	**On-Line Resources** **Teacherdeveloped notes**

			to solve software problems (e.g., design, coding, testing, verification). 3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. 7. Describe how various types of data are stored in a computer system. 8. Use modeling and simulation to represent and understand natural phenomena. 10. Describe the concept of parallel processing as a strategy to solve large problems. 11. Describe how computation shares features with art and music by translating human intention into an artifact. **5.3.A Computer Science in the Modern World - Collaboration** 1. Work in a team to design and develop a software artifact. 3. Describe how computing enhances traditional forms and enables new forms of experience, expression, communication, and collaboration. 4. Identify how collaboration influences the design and development of software products. **5.3.A Computer Science in the Modern World - Computing Practice and Programming** 3. Use various debugging and testing methods to ensure program correctness (e.g., test cases, unit testing, white box, black box, integration testing). 4. Apply analysis, design, and implementation techniques to solve problems (e.g., use one or more software lifecycle models). 5. Use Application Program Interfaces (APIs) and libraries to facilitate programming solutions.	**inheritance.** **Breaking down a task into smaller subtasks using methods.** **Creating enhanced Alice objects by adding new methods.** **Compare and contrast worldlevel and classlevel methods.** **Generalize Alice objects by adding parameters.** **Use different types of Alice parameters.** **Change the visibility of objects.**	**list?** **Why is it desirable to deconstruct a method into smaller submethods?** **What are the advantages to adding methods to existing Alice objects?** **What is the distinction between world-level and classlevel methods?** **Why is it necessary to use parameters instead of object names?** **What are the different types of Alice parameters?**	**Abstraction Argument By default Calling a method Class Class-level method Inheritance Instance Method Object Parameter Primitive method Stepwise refinement World-level method**	**Textbook**

			6. Select appropriate file formats for various types and uses of data.				

			7. Describe a variety of programming languages available to solve problems and develop systems. 8. Explain the program execution process. 12. Describe how mathematical and statistical functions, sets, and logic are used in computation. **5.3.B Computer Science Concepts and Practices - Computational Thinking** 4. Evaluate algorithms by their efficiency, correctness, and clarity. 8. Use models and simulations to help formulate, refine, and test scientific hypotheses. 10. Decompose a problem by defining new functions and classes. 11. Demonstrate concurrency by separating processes into threads and dividing data into parallel streams. **5.3.B Computer Science Concepts and Practices - Collaboration** 1. Use project collaboration tools, version control systems, and Integrated Development Environments (IDEs) while working on a collaborative software project. 3. Evaluate programs written by others for readability and usability. **5.3.B Computer Science Concepts and Practices - Computing Practice and Programming** 1. Use advanced tools to create digital artifacts (e.g., web design, animation, video, multimedia). 2. Use tools of abstraction to decompose a largescale computational problem (e.g.,		**What is an example of changing the opacity or visibility of an Alice object?**		

			procedural abstraction, object-oriented design, functional design). 3. Classify programming languages based on their				

			level and application domain **5.3.C.2 Projects-Based Courses**				

November	**Programming Events**	**Alice Events**	**5.3.A Computer Science in the Modern World - Computational Thinking** 1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. 2. Describe a software development process used to solve software problems (e.g., design, coding, testing, verification). 3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. 7. Describe how various types of data are stored in a computer system. 8. Use modeling and simulation to represent and understand natural phenomena. 10. Describe the concept of parallel processing as a strategy to solve large problems. 11. Describe how computation shares features with art and music by translating human intention into an artifact. **5.3.A Computer Science in the Modern World - Collaboration** 1. Work in a team to design and develop a software artifact. 3. Describe how computing enhances traditional forms and enables new forms of experience, expression, communication, and collaboration. 4. Identify how collaboration influences the design and development of software products.	**Use the keyboard to manipulate object movement during execution of an Alice animation.** **Evaluate the cause and effect of using Alice events in animation execution.** **Each time an event occurs, its corresponding event handling method is called.** **Parameters allow us to write one method that can handle several related events.**	**How can keyboard keys and a mouse click affect animation?** **How is an event created at runtime?** **What part of the Alice system is used to link an event to an event handling method?** **Why should you test an event handling method htat has a parameter several times, calling it with different parameters?**	**Participation** **Class Notes** **Quiz** **Presentation** **<u>Terms</u>** **Control of flow** **Event** **Event handling method** **Incremental development** **Interactive**	**On-Line Resources** **Teacherdeveloped notes** **Textbook**

			5.3.A Computer Science in the Modern World - Computing Practice and Programming				

			3. Use various debugging and testing methods to ensure program correctness (e.g., test cases, unit testing, white box, black box, integration testing). 4. Apply analysis, design, and implementation techniques to solve problems (e.g., use one or more software lifecycle models). 5. Use Application Program Interfaces (APIs) and libraries to facilitate programming solutions. 6. Select appropriate file formats for various types and uses of data. 7. Describe a variety of programming languages available to solve problems and develop systems. 8. Explain the program execution process. 12. Describe how mathematical and statistical functions, sets, and logic are used in computation. **5.3.B Computer Science Concepts and Practices - Computational Thinking** 4. Evaluate algorithms by their efficiency, correctness, and clarity. 8. Use models and simulations to help formulate, refine, and test scientific hypotheses. 10. Decompose a problem by defining new functions and classes. 11. Demonstrate concurrency by separating processes into threads and dividing data into parallel streams. **5.3.B Computer Science Concepts and Practices - Collaboration** 1. Use project collaboration tools, version control systems, and Integrated Development Environments (IDEs) while working on a collaborative software project.		**Why are events considered to be worldlevel in Alice?**		

			3. Evaluate programs written by others for readability and usability.				

			5.3.B Computer Science Concepts and Practices **- Computing Practice and Programming** 1. Use advanced tools to create digital artifacts (e.g., web design, animation, video, multimedia). 2. Use tools of abstraction to decompose a largescale computational problem (e.g., procedural abstraction, object-oriented design, functional design). 3. Classify programming languages based on their level and application domain **5.3.C.2 Projects-Based Courses**				

November - December	**Designing functions**	**Programmer -created functions and if/else**	**5.3.A Computer Science in the Modern World - Computational Thinking** 1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. 2. Describe a software development process used to solve software problems (e.g., design, coding, testing, verification). 3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. 7. Describe how various types of data are stored in a computer system. 8. Use modeling and simulation to represent and understand natural phenomena. 10. Describe the concept of parallel processing as a strategy to solve large problems. 11. Describe how computation shares features with art and music by translating human intention into an artifact. **5.3.A Computer Science in the Modern World - Collaboration**	**Relational operators can be used to compare values in a Boolean expression.** **Functions can be written to return a boolean value and used in if statements.** **Functions can be written to compute and return other types of values.**	**In a boolean expression containing multiple conditions, what operators may be used to connect the conditions?** **Why are additional functions needed in addition to the functions provided in the Alice programming language?**	**Participation** **Class Notes** **Quiz** **Presentation** **<u>Terms</u>** **Boolean expression** **Conditional execution** **Conditional expression** **Expression** **Function** **If/Else statement** **Integer** **Logical operator** **Random number** **Range**	**On-Line Resources** **Teacherdeveloped notes** **Textbook**

			1. Work in a team to design and develop a software artifact. 3. Describe how computing enhances traditional forms and enables new forms of experience, expression, communication, and collaboration. 4. Identify how collaboration influences the design and development of software products. **5.3.A Computer Science in the Modern World - Computing Practice and Programming** 3. Use various debugging and testing methods to ensure program correctness (e.g., test cases, unit testing, white box, black box, integration testing). 4. Apply analysis, design, and implementation techniques to solve problems (e.g., use one or more software lifecycle models). 5. Use Application Program Interfaces (APIs) and libraries to facilitate programming solutions. 6. Select appropriate file formats for various types and uses of data. 7. Describe a variety of programming languages available to solve problems and develop systems. 8. Explain the program execution process. 12. Describe how mathematical and statistical functions, sets, and logic are used in computation. **5.3.B Computer Science Concepts and Practices - Computational Thinking** 4. Evaluate algorithms by their efficiency, correctness, and clarity. 8. Use models and simulations to help formulate, refine, and test scientific hypotheses.		**Is it possible to have more than one type of value returned from a function?**	**Relational operator** **Return statement**	

			10. Decompose a problem by defining new functions and classes. 11. Demonstrate concurrency by separating processes into threads and dividing data into				

			parallel streams. **5.3.B Computer Science Concepts and Practices** **- Collaboration** 1. Use project collaboration tools, version control systems, and Integrated Development Environments (IDEs) while working on a collaborative software project. 3. Evaluate programs written by others for readability and usability. **5.3.B Computer Science Concepts and Practices** **- Computing Practice and Programming** 1. Use advanced tools to create digital artifacts (e.g., web design, animation, video, multimedia). 2. Use tools of abstraction to decompose a largescale computational problem (e.g., procedural abstraction, object-oriented design, functional design). 3. Classify programming languages based on their level and application domain **5.3.C.2 Projects-Based Courses**				

December	**Loops**	**Definite and Conditional Loops**	**5.3.A Computer Science in the Modern World - Computational Thinking** 1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. 2. Describe a software development process used to solve software problems (e.g., design, coding, testing, verification). 3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. 7. Describe how various types of data are stored in a computer system.	**Compare and contrast between counted and conditional loops.** **Loops can be nested within loops.** **The boolean**	**When is it advantageous to use a counted loop vs. a conditional loop?** **When is a nested loop necessary?**	**Participation** **Class Notes** **Quiz** **Presentation** **<u>Terms</u>** **Count** **Definite loop** **Indefinite loop** **Infinite loop** **Nested loop** **While**	**On-Line Resources** **Teacherdeveloped notes** **Textbook**

			8. Use modeling and simulation to represent and understand natural phenomena. 10. Describe the concept of parallel processing as a strategy to solve large problems. 11. Describe how computation shares features with art and music by translating human intention into an artifact. **5.3.A Computer Science in the Modern World - Collaboration** 1. Work in a team to design and develop a software artifact. 3. Describe how computing enhances traditional forms and enables new forms of experience, expression, communication, and collaboration. 4. Identify how collaboration influences the design and development of software products. **5.3.A Computer Science in the Modern World - Computing Practice and Programming** 3. Use various debugging and testing methods to ensure program correctness (e.g., test cases, unit testing, white box, black box, integration testing). 4. Apply analysis, design, and implementation techniques to solve problems (e.g., use one or more software lifecycle models). 5. Use Application Program Interfaces (APIs) and libraries to facilitate programming solutions. 6. Select appropriate file formats for various types and uses of data. 7. Describe a variety of programming languages available to solve problems and develop systems. 8. Explain the program execution process.	**condition used for a conditional loop is the same type as used in if statements.** **The count for a counted loop must be a positive whole number or infinity.** **When a loop is nested within a loop, the inner loop will fully execute each time the outer loop executes once.**	**How is the condition in a loop similar to a condition in an if/else statement?** **What happens if the count is a negative number in a loop?** **Does the order of nested loops being executed make a difference in the animation?**		

			12. Describe how mathematical and statistical functions, sets, and logic are used in computation.				

			5.3.B Computer Science Concepts and Practices - Computational Thinking 4. Evaluate algorithms by their efficiency, correctness, and clarity. 8. Use models and simulations to help formulate, refine, and test scientific hypotheses. 10. Decompose a problem by defining new functions and classes. 11. Demonstrate concurrency by separating processes into threads and dividing data into parallel streams. **5.3.B Computer Science Concepts and Practices - Collaboration** 1. Use project collaboration tools, version control systems, and Integrated Development Environments (IDEs) while working on a collaborative software project. 3. Evaluate programs written by others for readability and usability. **5.3.B Computer Science Concepts and Practices - Computing Practice and Programming** 1. Use advanced tools to create digital artifacts (e.g., web design, animation, video, multimedia). 2. Use tools of abstraction to decompose a largescale computational problem (e.g., procedural abstraction, object-oriented design, functional design). 3. Classify programming languages based on their level and application domain **5.3.C.2 Projects-Based Courses**				

January	**Computer Science**	**Computer Architecture**	**5.3.A Computer Science in the Modern World - Computing Practice and Programming**	**Identify and describe the**	**What is the purpose of**	**Participation Class Notes**	**On-Line Resources**

	Topics		10. Explore a variety of careers to which computing is central. **5.3.A Computer Science in the Modern World -** **Computers and Communications Devices** 2. Develop criteria for purchasing or upgrading computer system hardware. 3. Describe the principal components of computer organization (e.g., input, output, processing, and storage). 4. Compare various forms of input and output. 5. Explain the multiple levels of hardware and software that support program execution (e.g., compilers, interpreters, operating systems, networks). **5.3.A Computer Science in the Modern World -** **Community, Global, and Ethical Impacts** 1. Compare appropriate and inappropriate social networking behaviors. 9. Describe different ways in which software is created and shared and their benefits and drawbacks (commercial software, public domain software, open source development). **5.3.B Computer Science Concepts and Practices** **- Community, Global, and Ethical Impacts**	**four functions of a computer.** **Develop criteria for purchasing or upgrading computer system hardware.**	**owning and using a computer?** **What criteria are important when purchasing or upgrading a computer?**	**Quiz** **Presentation**	**Teacherdeveloped notes**

			7. Differentiate among open source, freeware, and proprietary software licenses and their applicability to different types of software.				
January	**Computer Science Topics**	**Computer History**	**5.3.A Computer Science in the Modern World - Computing Practice and Programming**	**Identify and describe the Four Generations of**	**What are the four generations of computer**	**Participation Class Notes Quiz Presentation**	**On-Line Resources Teacher-**

			10. Explore a variety of careers to which computing is central.				

			5.3.A Computer Science in the Modern World - Computers and Communications Devices 5. Explain the multiple levels of hardware and software that support program execution (e.g., compilers, interpreters, operating systems, networks). **5.3.A Computer Science in the Modern World - Community, Global, and Ethical Impacts** 1. Compare appropriate and inappropriate social networking behaviors. 9. Describe different ways in which software is created and shared and their benefits and drawbacks (commercial software, public domain software, open source development). **5.3.B Computer Science Concepts and Practices - Computing Practice and Programming** 6. Anticipate future careers and the technologies that will exist. **5.3.B Computer Science Concepts and Practices - Computers and Communications Devices** 1. Discuss the impact of modifications on the functionality of application programs. **5.3.B Computer Science Concepts and Practices - Community, Global, and Ethical Impacts** 2. Analyze the beneficial and harmful effects of computing innovations. 3. Summarize how financial markets, transactions, and predictions have been transformed by automation.	**Computer History, how they have defined Computer Science for today, and how they might predict the future for Computer Science.**	**history, and how do they affect you?** **What are the capabilities of a computer as time progressed?**		**developed notes**

			4. Summarize how computation has revolutionized the way people build real and virtual organizations				

			and infrastructures. 7. Differentiate among open source, freeware, and proprietary software licenses and their applicability to different types of software.				
January	**Computer Science Topics**	**Number Systems**	**5.3.A Computer Science in the Modern World -** **Computational Thinking** 5. Describe the relationship between binary and hexadecimal representations. **5.3.A Computer Science in the Modern World -** **Computers and Communications Devices** 5. Explain the multiple levels of hardware and software that support program execution (e.g., compilers, interpreters, operating systems, networks). **5.3.B Computer Science Concepts and Practices** **- Computational Thinking** 7. Discuss the interpretation of binary sequences in a variety of forms (e.g., instructions, numbers, text, sound, and image).	**Describe the importance of the relationship between binary, octal, and hexadecimal number systems.**	**What is the relationship that Computer Science has to binary, octal, and hexadecimal number systems?**	**Participation** **Class Notes** **Quiz** **Presentation**	**On-Line Resources** **Teacherdeveloped notes**

Optional Topic	**Scratch**	**Using the Scratch IDE**	**5.3.A Computer Science in the Modern World - Computational Thinking** 1. Use predefined functions and parameters, classes and methods to divide a complex problem into simpler parts. 2. Describe a software development process used to solve software problems (e.g., design, coding, testing, verification). 3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. 7. Describe how various types of data are stored in a computer system.	**Discover how to work with the components of Scratch to develop problem-solving skills and algorithmic thinking.** **Use in-class practice to**	**What are the components of Scratch that will facilitate solving a problem?** **What type of problems can be solved using the**	**Check during exploration time for progress and understanding.** **Check project assignments for implementation of Big Ideas and specific requirements.**	**On-Line Tutorials Self-guided exploration** **Teacherdeveloped practice** **Teacherdeveloped project**

			8. Use modeling and simulation to represent and understand natural phenomena. 10. Describe the concept of parallel processing as a strategy to solve large problems. 11. Describe how computation shares features with art and music by translating human intention into an artifact. **5.3.A Computer Science in the Modern World - Collaboration** 1. Work in a team to design and develop a software artifact. 3. Describe how computing enhances traditional forms and enables new forms of experience, expression, communication, and collaboration. 4. Identify how collaboration influences the design and development of software products. **5.3.A Computer Science in the Modern World - Computing Practice and Programming** 3. Use various debugging and testing methods to ensure program correctness (e.g., test cases, unit testing, white box, black box, integration testing). 4. Apply analysis, design, and implementation techniques to solve problems (e.g., use one or more software lifecycle models). 5. Use Application Program Interfaces (APIs) and libraries to facilitate programming solutions. 6. Select appropriate file formats for various types and uses of data. 7. Describe a variety of programming languages available to solve problems and develop systems. 8. Explain the program execution process.	**increase knowledge of basic programming conventions and syntax.** **Use specific Scratch project assignments to develop programming conventions and syntax.**	**specific Scratch programming components?** **How does Scratch differ from Alice?**		**assignments**

			12. Describe how mathematical and statistical functions, sets, and logic are used in computation.				

			5.3.B Computer Science Concepts and Practices **- Computational Thinking** 4. Evaluate algorithms by their efficiency, correctness, and clarity. 8. Use models and simulations to help formulate, refine, and test scientific hypotheses. 10. Decompose a problem by defining new functions and classes. 11. Demonstrate concurrency by separating processes into threads and dividing data into parallel streams. **5.3.B Computer Science Concepts and Practices** **- Collaboration** 1. Use project collaboration tools, version control systems, and Integrated Development Environments (IDEs) while working on a collaborative software project. 3. Evaluate programs written by others for readability and usability. **5.3.B Computer Science Concepts and Practices** **- Computing Practice and Programming** 1. Use advanced tools to create digital artifacts (e.g., web design, animation, video, multimedia). 2. Use tools of abstraction to decompose a largescale computational problem (e.g., procedural abstraction, object-oriented design, functional design).				

			3. Classify programming languages based on their level and application domain **5.3.C.2 Projects-Based Courses**				
Optional Topic	**Android APP Inventor**	**Using the Android App**	**5.3.A Computer Science in the Modern World -**	**Use in-class practice to**	**What are the components**	**Check during exploration**	**On-Line Tutorials**

			Computing Practice and Programming				
		Inventor IDE	2. Use mobile devices/emulators to design, develop, and implement mobile computing applications.	**increase knowledge of basic programming conventions and syntax.** **Use specific App Inventor project assignments to develop programming conventions and syntax.**	**of an Android project that will facilitate solving a problem?** **What type of problems can be solved using Android programming components?**	**time for progress and understanding.** **Check project assignments for implementation of Big Ideas and specific requirements.**	**Self-guided exploration** **Teacherdeveloped practice** **Teacherdeveloped project assignment s**

***** Some teachers may need to think about the assessments and resources used in order to determine the Big Ideas, Enduring Understandings, and Essential Questions embedded in their courses. At this point in your curriculum mapping, you might want to ignore the "Common Assessments" and "Common Resources Used" columns. However, you may use them if you wish.