Wyoming CS Standards Alignment with CodeX Curriculum

	Unit 1	Unit 2	Unit 3
Computing Systems			
8.CS.D.01 Recommend improvements to the design of computing devices based on an analysis of how a variety of users interact with the device.			
8.CS.HS.01 Design and refine a project that combines hardware and software components to collect and exchange data.			
8.CS.T.01 Systematically identify, resolve, and document increasingly complex software and hardware problems with computing devices and their components.	[1]		
Network and the Internet			
8.NI.NCO.01 Model the role of protocols in transmitting data across networks and the internet (e.g., explain protocols and their importance to data transmission; model how packets are broken down into smaller pieces and how they are delivered).			
8.NI.C.01 Critique physical and digital procedures that could be implemented to protect electronic data/information.			
8.NI.C.02 Apply multiple methods of encryption to model the secure transmission of data.			
Data Analysis			
8.DA.S.01 Represent data using multiple encoding schemes (e.g., ASCII, binary).			
8.DA.CVT.01 Using computational tools, transform collected data to make it more useful and reliable.			
8.DA.IM.01 Refine computational models based on generated data.			
Algorithms and Programming			
8.AP.A.01 Create flowcharts and pseudocode to design algorithms to solve complex problems.	[2]		
8.AP.V.01 Using grade appropriate content and complexity, create clearly named variables that represent different data types and perform operations on their values.	[3]		
8.AP.C.01 Using grade appropriate content and complexity, design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.	[4]		
8.AP.M.01 Using grade appropriate content and complexity, decompose problems and sub-problems into parts to facilitate the design, implementation, and review of programs.	[5]		
8.AP.M.02 Using grade appropriate content and complexity, create procedures with parameters to organize code and make it easier to reuse.		[6]	
8.AP.PD.01 Using grade appropriate content and complexity, seek and incorporate feedback from team members and users to refine a solution to a problem.			
8.AP.PD.02 Incorporate existing code, media, and libraries into original programs of increasing complexity and give attribution.	[7]		
8.AP.PD.03 Systematically test and refine programs using a range of test cases.			
8.AP.PD.05 Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.			
Impacts of Computing			
8.IC.C.01 Describe impacts associated with computing technologies that affect people's everyday activities and career options.			
8.IC.C.02 Describe issues of bias and accessibility in the design of technologies.			
8.IC.SI.01 Using grade appropriate content and complexity, collaborate using tools to connect with peers when creating a computational artifact.			

Wyoming CS Standards Alignment with CodeX Curriculum

	Unit 1	Unit 2	Unit 3
8.IC.SI.02 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.			
8.IC.SLE.01 Using grade appropriate content and complexity, describe tradeoffs between allowing information to be public and keeping information private and secure.			
8.IC.SLE.02 Using grade level appropriate content and complexity, discuss the legal, social, and ethical impacts associated with software development and use, including both positive and malicious intent.			

Wyoming CS Standards Alignment with CodeX Curriculum			
	Unit 1	Unit 2	Unit 3
Computing Systems			
L1.CS.D.01 Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.			
L1.CS.HS.01 Explain the interactions between application software, system software, and hardware layers.			
L2.CS.HS.01 Categorize the roles of operating system software.			
L1.CS.T.01 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and resolve errors.	[8]		
L2.CS.T.01 Identify how hardware components facilitate logic, input, output, and storage in computing systems, and their common malfunctions.			
Network and the Internet			
L1.NI.NCO.01 Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing.			
L2.NI.NCO.01 Describe the issues that impact network functionality (e.g., bandwidth, load, latency, topology).			
L1.NI.C.01 Give examples to illustrate how sensitive data can be affected by malware and other attacks.			
L2.NI.C.01 Compare ways software developers protect devices and information from unauthorized access.			
L1.NI.C.02 Recommend cybersecurity measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts.			
L1.NI.C.03 Compare various security measures, considering trade-offs between the usability and security of a computing system.			
L1.NI.C.04 Explain trade-offs when selecting and implementing cybersecurity recommendations.			
Data Analysis			
L1.DA.S.01 Translate between different bit representations of real-world phenomena, such as characters, numbers, and images.			
L1.DA.S.02 Evaluate the trade-offs in how data elements are organized and where data is stored.			
L1.DA.CVT.01 Create interactive data representations using software tools to help others better understand real-world phenomena (e.g., paper surveys and online data sets).			
L2.DA.CVT.01 Use data analysis tools and techniques to identify patterns in data representing complex systems.			
L2.DA.CVT.02 Select data collection tools and techniques, and use them to generate data sets that support a claim or communicate information.			
L1.DA.IM.01 Create computational models that represent the relationships among different elements of data collected from a phenomenon or process.			
L2.DA.IM.01 Formulate, refine, and test scientific hypotheses using models and simulations.			
Algorithms and Programming			
L1.AP.A.01 Create a prototype that uses algorithms (e.g., searching, sorting, finding shortest distance) to provide a possible solution for a real-world problem relevant to the student.			
L2.AP.A.01 Critically examine and trace classic algorithms. Use and adapt classic algorithms to solve computational problems (e.g., selection sort, insertion sort, binary search, linear search).			
L1.AP.A.02 Describe how artificial intelligence algorithms drive many software and physical systems.			
L2.AP.A.02 Develop an artificial intelligence algorithm to play a game against a human opponent or solve a real-world problem.			
L2.AP.A.03 Evaluate algorithms (e.g., sorting, searching) in terms of their efficiency, correctness, and clarity.			

Wyoming CS Standards Alignment with CodeX Curriculum			
	Unit 1	Unit 2	Unit 3
L1.AP.V.01 Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.		[9]	
L2.AP.V.01 Compare and contrast simple data structures and their uses (e.g., lists, stacks, queues).			
L1.AP.C.01 Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.			
L2.AP.C.01 Trace the execution of recursion, illustrating output and changes in values of named variables.	[10]		
L1.AP.C.02 Trace the execution of loops and conditional statements, illustrating output and changes in values of named variables.	[11]		
L1.AP.C.03 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.	[12]		
L1.AP.M.01 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.	[13]		
L2.AP.M.01 Construct solutions to problems using student-created components, such as procedures, modules, and/or objects.	[14]		
L1.AP.M.02 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.		[15]	
L2.AP.M.02 Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution.			
L2.AP.M.03 Demonstrate code reuse by creating programming solutions using libraries and APIs.	[16]		
L1.AP.PD.01 Plan and develop programs by analyzing a problem and/or process, developing and documenting a solution, testing outcomes, and adapting the program for a variety of users.			
L2.AP.PD.01 Plan and develop programs that will provide solutions to a variety of users using a software life cycle process.			
L1.AP.PD.02 Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.			
L2.AP.PD.02 Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (e.g., code documentation) in a group software project.			
L1.AP.PD.03 Use debugging tools to identify and fix errors in a program.	[17]		
L2.AP.PD.03 Develop programs for multiple computing platforms.			
L1.AP.PD.04 Design and develop computational artifacts, working in team roles, using collaborative tools.			
L2.AP.PD.04 Evaluate key qualities of a program through a process such as a code review (e.g., qualities could include correctness, usability, readability, efficiency, portability, and scalability).	[18]		
L1.AP.PD.05 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.	[19]		
L2.AP.PD.05 Develop and use a series of test cases to verify that a program performs according to its design specifications.			
L1.AP.PD.06 Evaluate and refine computational artifacts to make them more usable and accessible.	[20]		
L2.AP.PD.06 Explain security issues that might lead to compromised computer programs.			
L2.AP.PD.07 Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).			
L2.AP.PD.08 Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems.			
Impacts of Computing			
L1.IC.C.01 Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.			

Wyoming CS Standards Alignment with CodeX Curriculum			
	Unit 1	Unit 2	Unit 3
L2.IC.C.01 Evaluate the beneficial and harmful effects that computational artifacts and innovations have on society.			
L1.IC.C.02 Test and refine computational artifacts to reduce bias and equity deficits.			
L2.IC.C.02 Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society.			
L1.IC.C.03 Demonstrate how a given algorithm applies to problems across disciplines.			
L2.IC.C.03 Predict how computational innovations that have revolutionized aspects of our culture might evolve.			
L1.IC.SI.01 Use tools and methods for collaboration.			
L2.IC.SI.01 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.			
L1.IC.SI.02 Practice grade-level appropriate behavior and responsibilities while participating in an online community. Identify and report inappropriate behavior.			
L1.IC.SLE.01 Explain the beneficial and harmful effects that intellectual property laws can have on innovation			
L2.IC.SLE.01 Debate laws and regulations that impact the development and use of software and technology.			
L1.IC.SLE.02 Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users.			
L2.IC.SLE.02 Using grade level appropriate content and complexity, discuss the legal, social, and ethical impacts associated with software development and use, including both positive and malicious intent.			
L1.IC.SLE.03 Evaluate the social and economic implications of privacy in the context of safety, law, or ethics.			
L1.IC.SLE.04 Using grade level appropriate content and complexity, discuss the legal, social, and ethical impacts associated with software development and use, including both positive and malicious intent.			

[1] Mission 2 discusses troubleshooting techniques as does the teachers' manual

[2] These are introduced in the teachers' manual

[3] 3.8 begins the use of variables
5.5 discusses descriptive naming of variables

[4] Mission 6 begins the use of nested loops but does not discuss them
Mission 9 introduces compound conditionals

[5] These can be done with Code Tracing Charts

[6] This is discussed in Mission 7.6 and again in more detail in Mission 10.6

[7] All missions use libraries and when new ones are introduced they are explained.

[8] These can be accomplished with Code Tracing Charts that are introduced in teachers' manual

[9] 7.5 introduces the use of lists

[10] This can be done in the debugger or with flowcharts beginning with Mission 4

[11] This can be done in the debugger or with flowcharts beginning with Mission 4

[12] These are the remixes that are introduced in Mission 4

[13] Pseudocodes and Flowcharts are introduced in the teachers' manual

[14] These are the remixes that are introduced in Mission 4

[15] These are the remixes that are introduced in Mission 4

[16] Libraries are used in all missions and every time a new one is introduced, it is explained.

[17] 3.5 introduces the debugger

[18] Code Tracing Charts are introduced in the teachers' manual

[19] 5.5 introduces the use of comments

[20] Function use begins with Mission 4
Function creation begins with Mission 9