Skill Struck’s alignment to

Nevada Computer Science Standards

Legend

✅ = Standard aligned

🔷 = Not currently aligned

🚧 = Partially aligned (parts of the standard we do align with are highlighted yellow)

Standard	Status
1.1.1 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests	✅
1.1.2 Describe how artificial intelligence drives many software and physical systems	✅
1.1.3 Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem	✅
1.1.4 Use and adapt classic algorithms to solve computational problems	✅
1.1.5	✅

Develop classic algorithms in code to solve computational problems	
1.1.6 Evaluate algorithms in terms of their efficiency, correctness, and clarity	✅
1.2.1 Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made	✅
1.2.2 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions	✅
1.2.3 Illustrate the flow of execution of a recursive algorithm	✅
1.2.4 Implement conditional controls in code	✅
1.2.5 Implement recursive algorithms in code	✅
1.3.1 Demonstrate the use of both linked lists and arrays to simplify solutions, generalizing computational problems instead of repeatedly using simple variables	✅
1.3.2 Compare and contrast fundamental data structures and their uses	✅
1.3.3 Implement arrays in code	✅
1.3.4 Implement ArrayLists and LinkedLists in code	✅

1.4.1 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects	✅
1.4.2 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs	✅
1.4.3 Construct solutions to problems using student-created components, such as procedures, modules and/or objects	✅
1.4.4 Analyze a large-scale computational problem and identify generalizable patterns that can be applied to a solution	✅
1.4.5 Demonstrate code reuse by creating programming solutions using libraries and APIs	✅
1.5.1 Systematically design and develop programs for broad audiences by incorporating feedback from users	✅
1.5.2 Evaluate licenses that limit or restrict the use of computational artifacts when using resources such as libraries	✅
1.5.3 Evaluate and refine computational artifacts to make them more usable by all and accessible to people with disabilities	✅
1.5.4 Design and develop computational artifacts while working in team roles and using collaborative tools	✅

1.5.5 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs	✅
1.5.6 Plan and develop programs for broad audiences using a software life cycle process	✅
1.5.7 Explain security issues that might lead to compromised computer programs	✅
1.5.8 Develop programs for multiple computing platforms	✅
1.5.9 Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (code documentation) in a group software project	🚧
1.5.10 Develop and use a series of test cases to verify that a program performs according to its design specifications	✅
1.5.11 Modify an existing program to add additional functionality and discuss intended and unintended implications, e.g., breaking other functionality	✅
1.5.12 Evaluate key qualities of a program through a process such as a code review	✅
1.5.13 Compare multiple programming languages and discuss how their features make them suitable for solving different types of problems	✅

2.1.1 Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects	✅
2.2.1 Compare levels of abstraction and interactions between application software, system software, and hardware layers	✅
2.2.2 Categorize the roles of operating system software	✅
2.3.1 Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors	✅
2.3.2 Illustrate ways computing systems implement logic, input, and output through hardware components	✅
3.1.1 Translate between different bit representations of real-world phenomena, such as characters, numbers, and images, e.g., convert hexadecimal colors to decimal percentages, ASCII/Unicode representation	✅
3.1.2 Evaluate the tradeoffs in how data elements are organized and where data is stored	✅
3.1.3 Demonstrate the ability to store bit representation of real-world phenomena, characters, numbers, and images	✅
3.2.1 Create interactive data visualizations or alternative representations using software tools to help others better understand real-world phenomena	✅

3.2.2 Use data analysis tools and techniques to identify patterns in data representing complex systems	✅
3.2.3 Select data collection tools and techniques to generate data sets that support a claim or communicate information	✅
3.3.1 Create computational models that represent the relationships among different elements of data collected from a phenomenon, process, or model	✅
3.3.2 Evaluate the ability of models and simulations to test and support the refinement of hypotheses	✅
4.1.1 Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices	✅
4.1.2 Test and refine computational artifacts to reduce bias and equity deficits	✅
4.1.3 Demonstrate ways a given algorithm applies to problems across disciplines	✅
4.1.4 Explain the potential impacts of artificial intelligence on society	✅
4.1.5 Evaluate computational artifacts to maximize their beneficial effects and minimize harmful effects on society	✅
4.1.6	✅

Create computational artifacts to maximize their beneficial effects and minimize harmful effects on society	
4.1.7 Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society	✅
4.1.8 Predict how computational innovations that have revolutionized aspects of our culture might evolve	✅
4.2.1 Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields	✅
4.2.2 Use tools and methods for collaboration to increase the productivity of a team	✅
4.3.1 Explain the beneficial and harmful effects that intellectual property laws can have on innovation	✅
4.3.2 Explain the privacy concerns related to the collection and generation of data through automated processes that may not be evident to users	✅
4.3.3 Evaluate the social and economic implications of privacy in the context of safety, law, or ethics	✅
4.3.4 Discuss the role of ethics in emerging technologies	✅
4.3.5 Debate laws and regulations that impact the development and use of	✅

software	
5.1.1 Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing	✅
5.1.2 Describe the issues that impact network functionality, e.g., bandwidth, load, delay, topology	✅
5.2.1 Illustrate how sensitive data can be affected by malware and other attacks	✅
5.2.2 Recommend security measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts	✅
5.2.3 Compare various security measures, considering tradeoffs between the usability and security of a computing system	✅
5.2.4 Explain tradeoffs when selecting and implementing cybersecurity recommendations	✅
5.2.5 Compare ways software developers protect devices and information from unauthorized access	✅