## Computer Science Programming (NCTE):

The Computer Science Programming standards emphasize mastery of text-based programming. This is an advanced-level programming course. This is a non-career technical education course. These standards provide students the opportunity to utilize programming as a tool to construct well designed programs, with a focus on efficiency, user protections, and vulnerability mitigation. Through continued application of computational thinking and iterative design, students will address real-world problems by developing programming solutions. The standards focus on the enhancement of technical skills, including the integration of application programming interfaces, understanding Big O Notation, and proficiency in recursive algorithms.

### Algorithms and Programming (AP)

#### PRG.AP.1 The student will apply computational thinking to manage complex programs.

1. Identify and categorize real-world problems as classification, prediction, sequential decision, logical deduction, or statistical inference problem.
2. Analyze a large-scale computational problem, identify generalizable patterns, and implement a computing-based solution.
3. Decompose large-scale computational problems into subtasks and components processes and inter-relationships.
4. Implement and evaluate abstractions based on their modularity, reusability, and readability.

#### PRG.AP.2 The student will plan and implement programs that consist of compound conditionals, complex iterations, and complex computations using a text-based programming language.

1. Read and interpret algorithms expressed using plain language, and pseudocode. Read and write programs that include compound conditional execution and evaluate complex Boolean conditions.
2. Read and write programs that accept input from a variety of sources and produce output based on that input.
3. Read and write programs that include pre-defined and self-defined procedures.
4. Read and write programs that include functions with/without parameters, and functions with/without return values.
5. Read and write programs that consist of modular division, random number generation, substring manipulation and processing of individual characters.
6. Integrate external code with Application Programming Interface (APIs) and library calls.

#### PRG.AP.3 The student will use the iterative design process to create, test, and refine programs using a text-based programming language.

1. Trace the execution of iterative and recursive algorithms, illustrating output and changes in values of named variables.
2. Develop and systematically use a series of test cases to verify that a program performs according to its design specifications, including edge cases and all branches.
3. Use code review to evaluate the correctness, readability, and usability of a program.
4. Use debugging tools and user feedback to refine programs.
5. Modify existing program to improve functionality.

#### PRG.AP.4 The student will create programs that demonstrate an understanding of the data structures.

1. Use linear data structures: arrays, lists, and non-linear data structures.
2. Evaluate and convert data structures when appropriate.
3. Read and write programs that store, process, and manipulate 1D and 2D collections.
4. Identify how and when to use search and sort algorithms.
5. Read and write programs that include search and sort algorithms.

#### PRG.AP.5 The student will create programs that demonstrate an understanding of the interactions between classes and object-oriented design.

1. Define the role of inheritance, polymorphism, and encapsulation in object-oriented programming languages.
2. Use classes with instance data and methods to satisfy a design specification.
3. Organize programs methodically using comments and other organizational structures so that others can understand, interpret, and modify the program.

#### PRG.AP.6 The student will explain and justify program design and development decisions.

1. Explain the software life cycle and how it applies to the iterative design process.
2. Justify and communicate decisions and design elements.

#### PRG.AP.7 The student will interpret, adapt, test, debug, and refine algorithms for use in a particular context and evaluate for efficiency.

1. Use Big O notation to compare the benefits and drawbacks of using different algorithms for a particular process.

### Computing Systems (CSY)

#### PRG.CSY.1 The student will evaluate the relationship between storage, processing, and efficiency, and analyze the role in program development.

1. Create programs that utilize persistent storage for program input and output.
2. Define the role of cache memory.
3. Analyze the impact of different types of memory on program processing speed.
4. Conduct a cost-benefit analysis for different types of memory.
5. Redesign a program to improve efficiency and performance.

### Cybersecurity (CYB)

#### PRG.CYB.1 The student will evaluate current and emerging programming security practices.

1. Create programs that safeguard against user error.
2. Create programs that implement encryption algorithms.
3. Describe how software programs can meet basic requirements for security based on best practices.
4. Describe the impact of software vulnerabilities.
5. Evaluate methods developers use to protect unauthorized access to programs.

#### PRG.CYB.2 The student will write or adapt a program to avoid common vulnerabilities.

1. Understand the role of input validation in programming.
2. Develop code that validates input based on defined specifications.
3. Explain common vulnerabilities in program function and their impact.
4. Understand the impact of vulnerabilities on program function and security.

### Data and Analysis (DA)

#### PRG.DA.1 The student will evaluate the tradeoffs between a variety of data organization and storage options.

1. Identify and compare data organization methods: variables, arrays, lists, trees, and schemas.
2. Assess and compare data storage options such as databases, file systems, local storage, and cloud storage, for scalability, reliability, privacy, and cost.
3. Evaluate the impact of data organization and storage choices on program performance, efficiency, and resource utilization.

#### PRG.DA.2 The student will use a variety of data types and structures in representing programmatic solutions to real-world problems.

1. Research and describe real-world reasoning problems that a reasoning algorithm can be used to sort data.
2. Read data summaries and visualizations and explain/translate into nontechnical terms for various audience groups.
3. Collect, use, and manipulate data from a variety of types and structures.
4. Utilize data analysis to create programmatic solutions and draw conclusion based on the results.

#### PRG.DA.3 The student will identify data biases in the data collection process and describe privacy concerns surrounding data collection and processing.

1. Use the data cycle in the collection and processing of data as part of the development of a program.
2. Describe how the data collection process should be focused, relevant, and limited to the scope of the project.
3. Analyze data to identify outliers or missing variables that could result in data biases.
4. Describe privacy considerations in the collection of data.

#### PRG.DA.4 The student will use a programming language to develop a data visualization.

1. Identify libraries and other resources that enable the visualization of data inputs.
2. Compare and contrast the methods of creating data visualizations, including programming languages and application software.
3. Develop a data visualization using a programming language’s data processing function.
4. Create visualizations for descriptive and inferential statistical analysis based on the context and intended audience.
5. Apply mathematical operations and algorithms to manipulate and extract insights from data sets.
6. Justify the design, use, and effectiveness of different forms of data visualizations.

### Impacts of Computing (IC)

#### PRG.IC.1 The student will examine the ramifications of technical and ethical design decisions when developing applications.

1. Use a design document to explain the reasoning for the design decisions made when developing an application.
2. Research the effects of technical design decisions on overall program function.
3. Examine and explain the impacts of unintended consequences related to program design.

#### PRG.IC.2 The student will use data to analyze the impact of screen time on one’s mental and physical health.

1. Use statistical data to analyze the relationship between excessive screen time and attention span.
2. Analyze screen time usage data and propose recommendations to promote healthy habits.
3. Examine and discuss the impact of screen time and social media on academic or workplace performance.

#### PRG.IC.3 The student will expand computer science career explorations with work-based learning experiences.

1. Engage in work-based learning experiences involving computer science.

### Networks and the Internet (NI)

No additional standards for Networks and the Internet.