COMPUTER PROGRAMMING 1, 2, ADVANCED COMPUTER PROGRAMMING
(NEW NAMES: INRODUCTION TO COMPUTER PROGRAMMING, INTERMEDIATE COMPUTER PROGRAMMING, ADVANCED COMPUTER PROGRAMMING 2022-23)
COURSE CODE: 5050, 5051, 5376
As aligned to:
South Carolina Computer Science High School Process and Content Standards

The South Carolina Computer Science and Digital Literacy Process Standards should be integrated into every grade level within the South Carolina Computer Science and Digital Literacy Content Standards. Because the Process Standards drive the pedagogical component of teaching and serve as the means by which students should demonstrate understanding of the content standards, the process standards must be incorporated as an integral part of overall student expectations when assessing content understanding.

A computer science literate student can:

1. Foster an inclusive computing culture.
   a. Recognize that equitable access to computing benefits society as a whole.
   b. Consider others' perspectives as well as one's own perspective when developing computational solutions.
   c. Consider the needs of a variety of end users regarding accessibility and usability.

2. Collaborate around computing.
   a. Select appropriate technological tools that can be used to collaborate on a project.
   b. Collaborate productively with individuals of varying perspectives, skills, and backgrounds.
   c. Set and implement equitable expectations and workloads when working in teams.
   d. Integrate constructive feedback while working in teams.

3. Recognize, define, and analyze computational problems.
   a. Recognize when it is appropriate to solve a problem computationally.
   b. Make sense of computational problems and persevere in solving them.
   c. Relate computational problems to prior knowledge.
   d. Recognize that there may be multiple approaches to solving a problem.
   e. Approach problem solving iteratively, using a cyclical process.

4. Create, test, and refine computational artifacts.
   a. Consider the purpose of computational artifacts for practical use, personal expression, and/or societal impact.
   b. Recognize when to use the same solution for multiple problems.
   c. Test computational artifacts systematically by considering multiple scenarios and using test cases.
   d. Approach troubleshooting systematically.

e. Consider performance, reliability, usability, and accessibility when evaluating and refining computational artifacts.

5. Communicate about computing.
   a. Select and use appropriate technological tools to convey solutions to computing problems.
   b. Communicate about computational processes and solutions using appropriate terminology consistent with the intended audience and purpose.
   c. Articulate ideas responsibly by observing intellectual property rights and giving appropriate attribution.

South Carolina Computer Science High School Content Standards

Computing Systems
Standard 1: Examine how hardware and software contribute to computing devices solving relevant problems.
HS2.CS.1.1
Investigate how a problem is systematically solved through the selection and integration of hardware and software components.
HS3.CS.1.1
Recommend modifications for existing computing devices and software to improve functionality for end users.
HS4.CS.1.1
Develop a solution to a given problem using appropriate hardware and software (e.g., sensor devices, Wi-Fi capabilities, specialized displays, runtime modules, operating systems, application programming interfaces (APIs)).

HS1.CS.1.2
Compare and contrast the elements of a computing system by examining hardware elements for their intended use (e.g., input-output (I/O) devices, random access memory (RAM), read only memory (ROM), storage devices, motherboards, and processors including the arithmetic logic unit (ALU), control unit, registers, cache memory, example implementations of some of these components using logic gates) (Virginia, 2017).
HS2.CS.1.2
Analyze how various hardware and software layers provide simplifying abstractions (e.g., a redundant array of independent disks (RAID) controller hiding details of data storage on multiple disks, an operating system hiding details of virtual memory, the presentation layer of a network hiding details about encryption).
HS3.CS.1.2
Justify hardware and software selections for specific applications by evaluating the components (e.g., databases, sensors, application programming interfaces (APIs)) of various computing devices (e.g., desktops, laptops, tablets, smartphones, specialized devices like global positioning systems (GPSs)).
HS4.CS.1.2
Cite evidence of how selecting appropriate hardware and software components enhances user interfaces to provide better solutions for real-world problems.

HS4.CS.1.3
Defend the choice of an appropriate operating system based on the requirements of a given computer system or project.

Standard 2: Troubleshoot common hardware and software problems.
HS4.CS.2.1
Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors (CSTA, 2017).

Networks and the Internet
Standard 2: Evaluate cybersecurity threats and appropriate security measures across networks.
HS2.NI.2.1
Evaluate how sensitive data can be affected by malware and other attacks (e.g., denial-of-service attacks, ransomware, viruses, worms, spyware, phishing) (CSTA, 2017).
HS4.NI.2.1
Recommend security measures (i.e., hardware, software, and practices that control access to data and systems) to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts (CSTA, 2017).

HS1.NI.2.2
Identify best practices of software development that improve computer security and protect devices and information from unauthorized access (e.g., encryption, authentication strategies, secure coding, safeguarding keys) (CSTA, 2017).
HS2.NI.2.2
Compare and contrast ways software developers protect devices and information from unauthorized access (e.g., encryption, authentication strategies, secure coding, safeguarding keys) (CSTA, 2017).
HS3.NI.2.2
Evaluate various security measures, considering tradeoffs between the usability and security of a computing system. (e.g., a web filter that prevents access to many educational sites but keeps a campus' network safe) (CSTA, 2017).
HS4.NI.2.2
Select and justify cybersecurity recommendations (i.e., hardware, software, and practices that control access to data and systems) appropriate for an intended audience and purpose (CSTA, 2017).

Data and Analysis
Standard 1: Evaluate various data collection methods, data storage tools, data analysis tools, data representation tools, and bit representation.
HS1.DA.1.1
Describe the various data collection methods, data analysis tools, and data representation tools.
HS2.DA.1.1
Compare and contrast the various data collection methods, data analysis tools, and data representation tools.

HS3.DA.1.1
Explain how different collection methods and tools influence the amount and quality of the data that is observed and recorded.
HS4.DA.1.1
Justify the choice of a data collection method, data analysis tool, and data representation tool over alternate options.

HS1.DA.1.2
Describe the various data storage tools and data organization methods.
HS2.DA.1.2
Compare and contrast the various data storage tools and data organization methods.
HS3.DA.1.2
Justify choices on how data elements are organized and where data is stored considering cost, speed, reliability, accessibility, privacy, and integrity (e.g., local storage, portable storage, cloud storage).
HS4.DA.1.2
Evaluate the data storage needs of a computing solution (e.g., file compression).

HS1.DA.1.3
Distinguish between various methods of data representation (i.e., analog, digital, binary).
HS2.DA.1.3
Translate between various methods of data representation (i.e., analog, digital, ASCII, binary).

Standard 2: Construct a computational model using large data sets of real-world phenomenon.
HS1.DA.2.1
Describe the properties of a data set that could be used to explore a real- world phenomenon or support a claim.
HS2.DA.2.1
Compare and contrast data sets that could be used to explore a real-world phenomenon or support a claim.
HS3.DA.2.1
Create data sets that could be used to explore a real- world phenomenon or support a claim.
HS4.DA.2.1
Evaluate the use of large data sets to explore a real- world phenomenon or support a claim.

Standard 3: Create various ways to visually represent data.
HS1.DA.3.2
Organize collected data to communicate the solution to a real-world phenomenon and support a claim.

Algorithms and Programming
Standard 1: Design algorithms that can be adapted to express an idea or solve a problem.

HS1.AP.1.1
Create flowcharts and/or pseudocode to express a problem or idea as an algorithm.
HS2.AP.1.1
Create algorithms to solve computational problems that have an application in the real world (e.g., local community, church, civic organization, school, home life).
HS3.AP.1.1
Adapt predefined algorithms to solve computational problems.
HS4.AP.1.1
Evaluate algorithms in terms of efficiency, correctness, and clarity (CSTA, 2017).
Standard 2: Build a combination of control structures that supports complex execution, readability, and program performance.
HS1.AP.2.1
Trace the flow of execution of a program that uses a combination of control structures (e.g., conditionals, loops, event handlers, recursion).
HS2.AP.2.1
Design and iteratively develop programs that combine control structures (e.g., conditionals, loops, event handlers, recursion).
HS3.AP.2.1
Justify the selection of specific control structures explaining the benefits and drawbacks of choices made (e.g., tradeoffs involving implementation, readability, and program performance).

HS1.AP.2.2
Trace the flow of execution of a program that uses a variety of programming constructs (e.g., procedures, modules, objects).
HS2.AP.2.2
Design a solution through systematic analysis using programming constructs (e.g., procedures, modules, objects).
HS3.AP.2.2
Justify the selection of specific programming constructs, explaining the benefits and drawbacks of choices made on the program's execution.

Standard 3: Divide a task into sets of functional units that can be reused to compose a complex solution.
HS1.AP.3.1
Decompose tasks into smaller, reusable parts to facilitate the design, implementation, and review of programs.
HS3.AP.3.1
Build a complex solution to a problem that incorporates reusable code (e.g., student created, application programming interfaces (APIs), libraries).
HS4.AP.3.1
Justify the selection of modular parts in the creation of a complex solution.

Standard 4: Plan, build, test, refine, and document programs using text-based coding languages to solve problems with varying degrees of difficulty

HS1.AP.4.1
Plan and develop programs for a variety of audiences using a process that incorporates development, feedback, and revision.
HS2.AP.4.1
Plan and develop a program that addresses potential security issues.
HS3.AP.4.1
Plan and develop a program that is accessible across multiple computing platforms (e.g., iOS, Unix, Windows, web-based).
HS4.AP.4.1
Evaluate a program through a review process (e.g., code review, beta testing, pilot group).

HS1.AP.4.2
Seek and incorporate feedback to refine a solution (e.g., users, team members, code review, teachers).
HS2.AP.4.2
Systematically test programs using a range of test cases to meet design specifications (e.g., specific outcomes, functionality, user interface, error handling) (CSTA, 2017).
HS3.AP.4.2
Evaluate and refine programs to make them more usable, functional, and accessible.
HS4.AP.4.2
Implement version control to track program refinements.

HS1.AP.4.3
Recognize the variety of documentation methods available while developing a program (e.g., inline comments, procedure header, purposeful naming).
HS2.AP.4.3
Document programs in order to make them easier to follow, test, and debug.
HS3.AP.4.3
Document programs that use non-user-created resources (e.g., code, media, libraries) giving attribution to the original creator.
HS4.AP.4.3
Justify design decisions by documenting the design process of complex programs (e.g., developer journal, digital portfolio, presentation).

HS1.AP.4.4
Examine licenses (i.e., permissions) that limit or restrict use of resources (e.g., freeware, shareware, open source, creative commons).
HS2.AP.4.4
Discuss the implications of using licensed resources in a developed solution.

HS3.AP.4.4
Develop a systematic solution that incorporates licensed resources.
HS4.AP.4.4
Research the process for licensing student-created resources.

Standard 5: Choose data types and data structures based on functionality, storage, and performance tradeoffs.
HS1.AP.5.1
Justify and use appropriate data types (i.e., primitive and non-primitive) in simple programs.
HS2.AP.5.1
Determine when data structures (e.g., lists, arrays, tuples, stacks, queues, structures) are more appropriate than simple data types and incorporate them in programs.
HS3.AP.5.1
Determine when external data structures (e.g., databases, flat files) are appropriate and incorporate them in programs.
HS4.AP.5.1
Justify how data structures and abstraction are used to manage program complexity.

Impact of Computing
Standard 1: Evaluate the impact of computing from a global perspective.
HS1.IC.1.1
Research computing solutions to problems in different countries, considering the personal, ethical, social, economic, and cultural impact (e.g., the use of drones to deliver blood and medical supplies in countries in Africa, the use of Uber in India to address traffic congestion).
HS2.IC.1.1
Compare and contrast the efficiency, feasibility, and ethical impacts of deploying the same computing solution in various countries.

Standard 3: Understand the importance of access and equity in computing.
HS1.IC 3.3
Identify the advantages and disadvantages of diverse perspectives and backgrounds when solving computational problems.
HS2.IC.3.3
Evaluate existing computing solutions according to inclusivity or non-inclusivity (e.g., sight and hearing impairment, ethnicity, age).
HS3.IC.3.3
Recommend modifications to make a current computing solution more inclusive for all users.