Instructional Framework

SOFTWARE AND APP DESIGN
15.1200.40

Domain 1: Programming **Instructional Time: 50-60%**	
STANDARD 4.0 UTILIZE PRIMITIVE DATA TYPES AND STRINGS IN WRITING PROGRAMS	
4.1 Declare numeric Boolean, character, string variables, and float and double	● Various data types ● Usage of the data types ● Storage size and value ranges of the data types
4.2 Choose the appropriate data type for a given situation	● Data type requirement
4.3 Identify the correct syntax and usage for constants and variables in a program	● Naming Conventions ● Difference between the both and usage
4.4 Identify the correct syntax and safe functions for operations on strings, including length, substring and concatenation	● String operations ● Existing string functions usage/limitations
4.5 Explain complications of storing and manipulating data, i.e the Big-O notation for analyzing storage and efficiency	● Memory usage ● Complexity algorithms
4.6 Research industry relevant programming languages, i.e Java, JavaScript, and Python	● High/Low level languages - Basic information ● Basics of industry relevant languages; ex. Java/Python
STANDARD 5.0 PERFORM BASIC COMPUTER MATHEMATICS IN INFORMATION TECHNOLOGY	
5.1 Apply basic mathematics to hardware (e.g. bits, bytes, kilobytes, megabytes, gigabytes, terabytes)	● Various units of digital storage
5.2 Use binary to decimal, decimal to hexadecimal, hexadecimal to decimal, binary to hexadecimal, and binary to hexadecimal conversions to solve hardware and software problems	● Knowledge of various number systems ● Number system conversion
5.3 Identify and correctly use arithmetic operations applying the order of operations (precedence) with respect to programming	● Understand Arithmetic operators (+,-,*,/, %) ● Order of operations(PEMDAS/BODMAS acronyms)
5.4 Interpret and construct mathematical formulas	● Problem understanding and equation formation
5.5 Identify correct and problematic uses of integers, floating-point numbers, and fixed-point numbers in arithmetic	● Difference between numeric data types ● Data type usage/limitations
STANDARD 6.0 UTILIZE CONDITIONAL STRUCTURES IN WRITING PROGRAMS	
6.1 Use the correct syntax for decision statements (e.g. if/else, if, switch case)	● Controlling program flow based on various conditions ● Usage of if/ if-else/switch case statements
6.2 Compare values using relational operators (e.g. =,<,>, <=, >=, not equal)	● Various conditional operators and their results ● Difference between inclusive/exclusive limits(> and >=)
6.3 Evaluate Boolean expressions (e.g. AND, OR, NOT, NOR, XOR)	● Various Boolean expression evaluation ● DeMorgan's Law

	● Short-Circuit(McCarthy) evaluation
6.4 Use the correct nesting for decision structures	● Nested if/if-else ● Hierarchy of Conditional Check
STANDARD 7.0 UTILIZE ITERATIVE STRUCTURES IN WRITING PROGRAMS	
7.1 Identify various types of Iteration structure(e.g. while, for, for-each, recursion)	● Various types of iterations ● Usage of iterative types
7.2 Identify how loops are controlled (various conditions and exits)	● Break keyword usage ● Control variable ● Increment/decrement control variable
7.3 Use the correct syntax for nested loops	● Construct nested loop. ● Control the inner loops through the outer loop
7.4 Compute the values of variables involved with nested loops	● Code tracing ● Working with the iteration variables and how they control each others value
STANDARD 8.0 UTILIZE BASIC DATA STRUCTURES IN WRITING PROGRAMS	
8.1 Demonstrate basic uses of arrays including initialization, storage, and retrieval of values	● Array ● Types of arrays(e.g. integer/String) ● Indices of array elements
8.2 Distinguish between arrays and hashmaps	● Array values v/s a hashmap has key and values
8.3 Identify techniques for declaring, initializing, and modifying user-defined data types	● Appending arrays and arraylists ● Class type arraylists
8.4 Search and sort data in an array	● Search and Sort algorithms; ex. Bubble, Linear, Binary
8.5 Create and use two-dimensional arrays	● Need/use of 2D arrays ● Concept of rows and columns ● Format and access the elements of 2D array(a[][])
8.6 Describe the efficiency of different sorting algorithms (e.g. bubble, insertion, merge)	● Processing Time/memory usage with different algorithms ● Best and worst case scenarios
STANDARD 14.0 UTILIZE AND CREATE COMMUNITY RESOURCES	
14.1 Use standard library functions	● Download and import standard libraries relevant to the programming language in use
14.2 Find and use third party libraries (e.g. web-based, package managers)	● Various free/paid reusable components available for the said language; ex. SDK in Java
14.3 Explain and interact with an Application Program Interface (API)	● Various software components interact with each other in programming
STANDARD 15.0 USE VERSION CONTROL SYSTEMS	
15.1 Identify the purpose of version control systems (e.g. Git, Mercurial)	● Version control ● Tracking needs
15.2 Create a new repository	● Central location for data/program storage ● Location option; ex. local/central/cloud-based

15.3 Add, push, and pull source code from repository	● Add to repository ● Push and pull source code
15.4 Explain branching and its uses	● Duplicate a program code for revision control purpose ● Parallel modifications between various team members
15.5 Restore previous versions of code from the repository	● Use of 'Reset' and other options available to restore previous version ● Pros/cons of backing up
STANDARD 18.0 EMPLOY OBJECT-ORIENTED PROGRAMMING TECHNIQUES	
18.1 Make a distinction between an object and a class	● Concept of Object Oriented Programming ● Object ; e.g. Dog is an object of class Animal
18.2 Differentiate among inheritance, composition and class relationships	● Is-a and has-a relationships ● Inherited/reused by a child class from parent class
18.3 Instantiate objects from existing classes	● Syntax of object declaration, initiation
18.4 Read the state of an object by invoking accessor methods	● Getter methods. e.g. getStudentName().
18.5 Change the state of an object by invoking a modifier method	● Setter methods. e.g. setStudentName("name")
18.6 Determine the requirements for constructing new objects by reading the documentation	● Single or multiple objects for a single class ● e.g. Multiple objects for a database creation
18.7 Create a user-defined class	● User defined data type and methods
18.8 Create a subclass of an existing class	● Subclass to a superclass ● Subclass specific data/methods only and reuse superclass methods
18.9 Identify the use of an abstract class as opposed to an interface	● Interface vs. abstract class ● Multiple inheritance
18.10 Explain the object-oriented concepts of polymorphism, inheritance, and encapsulation	● Reuse of same method names for different behavior/actions ● Subclasses to inherit the existing code/methods ● Accessor types(public/private/protected) to secure the data

Domain 2: App Design

Instructional Time: 10-20%

STANDARD 11.0 DEMONSTRATE PROGRAM ANALYSIS AND DESIGN	
11.1 Implement the steps in the System Development Life Cycle (SDLC) (e.g. planning, analysis, design, development, testing, implementation, maintenance)	● SDLC model; ex. waterfall, Spiral, Agile ● SDLC model choices ● Stages of the life cycle (requirements/documents/expectations/duration)
11.2 Develop program requirements/specifications and a testing plan (e.g. user stories, automated testing, test procedures)	● Client requirements ● Creating requirement documents ● Test cases needed for the testing phase
11.3 Apply pseudocode or graphical representations to plan the structure of a	● Pseudocode/flowcharts/algorithms

program or module (e.g. flowcharting, white boarding, UML)	● Collaborative planning; ex. flowcharting/white boarding, or other unified modeling language
11.4 Create and implement basic algorithms	● Good algorithm features ● Inputs/process/outputs/finite flow/performance of an algorithm ● Algorithm into programming language
STANDARD 12.0 DEVELOP A PROGRAM	
12.1 Use a program editor to enter and modify code	● IDE, MIT App Inventor, Greenfoot, Textpad, BlueJ, Eclipse
12.2 Identify correct input/output statements	● Programming language syntaxes ● Case sensitive languages errors
12.3 Choose the correct method of assigning input to variables including data sanitization	● Appropriate data types to variable assignment(s) ● Conditional checks ● Exception handling
12.4 Choose the correct method of outputting data with formatting and escaping	● Language specific output statements and escape characters ● Usage formatting data
12.5 Differentiate between interpreted and compiled code (e.g. steps necessary to run executable code)	● Interpreted code vs. compiled code ● Interpreted code/user language ● Compiled code/machine language ● Byte code/OOP languages
12.6 Apply industry standards in documentation (e.g. self-documenting code; function-level, program-level, user-level documentation)	● Document(with the code, within the code) ● Comments in code - third party understanding ● Documentation levels; ex. Self, Program, Function, User
12.7 Name identifiers and formatting code by applying recognized conventions	● Naming conventions and practices in programming (camelCase) ● Indentations and whitespace to format code
12.8 Demonstrate refactoring techniques to reduce repetitious code and improve maintainability	● Methods/functions to include blocks of code that can be reused ● Call/access these methods for code reusability
12.9 Demonstrate the use of parameters to pass data into program modules	● Parameters ● Method call ● Datatype of parameters ● Arguments data type in module definition
12.10 Demonstrate the use of return values from modules	● The module processes data/parameters and should return some kind of data/output. ● Use of return statements. ● How to catch the return values and use them further. ● Datatype of returning value to match exactly with return data type in the module definition
STANDARD 13.0 TEST AND DEBUG TO VERIFY PROGRAM OPERATION	
13.1 Identify errors in program modules	● Rectify syntax errors while writing the program ● IDEs that highlight/suggest these errors
13.2 Identify boundary cases and generate appropriate test data	● Limits(upper and lower) for various data/variables ● Test cases to address these limits

13.3 Perform integration testing including tests within a program to protect execution from bad input or other run-time errors	● Test groups of code to avoid errors ● Unit vs. integration testing strategies
13.4 Categorize, identify, and correct errors in code, including syntax, semantic, logic and runtime	● Errors in code ● Techniques for debug errors
13.5 Perform different methods of debugging (e.g. hand-trace code or real time debugging tools)	● Debugging techniques; ex. Interactive, print, remote ● Hand trace code or use a real time debugging tool; ex.EMPL
STANDARD 17.0 USE AND UPDATE DATA STORAGE AND MANAGEMENT	
17.1 Input/output data from a sequential file or database(DB)	● Database management and storage ● Basics of procedural languages; ex. PL/ SQL ● Access and modify data in a DB file
17.2 Demonstrate creating, reading, updating, and dropping a database	● Simple database ● Duplicate/incorrect entries
17.3 Demonstrate the proper use of SQL database applications that work with different languages.	● SQL applications; ex. MongoDB, Microsoft Access, Oracle Database, code.org’s App Lab
STANDARD 19.0 EMPLOY RUN TIME AND ERROR HANDLING TECHNIQUES	
19.1 Identify run time errors	● Runtime error vs. syntax errors. ● Invalid input
19.2 Describe error handling strategies	● Response and recovery procedure ● Cause/source of the runtime error from it’s name ● Source of the error, identify and fixing strategies for the line of code/module responsible
19.3 Handle unexpected return values	● Employ Conditional checks/boundaries
19.4 Handle (catch) runtime errors and take appropriate action	● Appropriate error message for unexpected returns values ● Alternative steps with try- catch-handle blocks ● Susceptible module within the try block
19.5 Throw standard exception classes	● Standard exceptions that occur at runtime ● Standard exception handling classes
19.6 Develop and throw custom exception classes	● User defined exception handling class ● Custom exception handling program

Domain 3: Computer Principles

Instructional Time: 5-10%

STANDARD 1.0 APPLY PROBLEM-SOLVING AND CRITICAL THINKING SKILLS	
1.1 Explain objectives and outcomes for a task.	● Problem objectives and the solution(s) required
1.2 Explain the process of decomposing a large programming problem into more smaller, more manageable procedures.	● Divide and conquer approach of problem solving ● Independent modules of the problem solution (Agile methodology)

1.3 Explain”visualizing” as a problem-solving technique prior to writing code	● Backwards planning technique ● Pseudo-code
1.4 Describe problem-solving and troubleshooting strategies applicable to software development.	● Problem debugging, rigorous test cases, user acceptance testing
STANDARD 2.0 RECOGNIZE SECURITY ISSUES	
2.1 Identify common computer threats.	● Types of computer threats ● Types of malwares ex. worms, viruses, trojan horses,spamming,Identity theft ● Protection against the malwares ● Cyber Safety
2.2 Describe potential vulnerabilities in software.	● Open Web Application Security Project (OWASP) to identify web vulnerability ● Scripting, authentication and injections as vulnerabilities in software
2.3 Identify procedures to maintain data integrity and security	● Personal safety precautions / best practices; ex. password protected systems/application. ● End user computer security risks
2.4 Explain best practices to maintain integrity and security in software development.	● Integrity and security in software ● Encryption/decryption ● Limited user access group and use of encryption algorithms/keys; ex. RSA
2.5 Describe methods for sanitizing user input to prevent issues.	● Error handling techniques; ex. try-catch blocks ● Checks/conditions to avoid runtime errors due to invalid user input ● Buffer overflows
2.6 Explain the CIA (confidentiality, integrity, and availability) triad.	● Model to guide policies of security information ● Role each element plays in the security plan development process
2.7 Explain how Software defects relate to software security.	● Phishing/ untrusted scripts from trusted sources ● Buffer overflows and cross site scripting
STANDARD 3.0 EXAMINE LEGAL AND ETHICAL ISSUES RELATED TO INFORMATION TECHNOLOGY	
3.1 Explore intellectual property rights including software licensing and software duplication [e.g. Digital Millennium Copyright Act (DMCA), software licensing, software duplication]	● Copyrights and citation standards and regulations ● Creative commons and other organizations that issue copyright licenses

3.2 Compare and contrast open source and proprietary systems in relations to legal and ethical issues (e.g. data pricing, use of public and private networks, social networking, industry-related data, data piracy)	● Fair use doctrine ● Digital Information literacy and citizenship ● Royalty fee, protection of author's work
3.3 Identify issues and regulations affecting computers, other devices, the internet , and information privacy (e.g. HIPAA, COPPA, CISPA, FERPA, PCI, GDPR, data brokers)	● Federal regulations vs. personal privacy online and devices

Domain 4: Web Design

Instructional Time : 5-10%

STANDARD 9.0 IDENTIFY INTERNET PROTOCOLS AND OPERATIONS	
9.1 Explain cloud-based computing and content delivery networks(CDN)	● Cloud-based computing ● Applications for use ● Cookies and cache in the settings ● CDN caches/data locally
9.2 Identify the components and functions of the internet(e.g. HTTP, HTTPS, FTP, IP addresses, IMAP)	● Basic Internet terminology ● OSi model ● Layers of API and frameworks ● Secure vs insecure browsing
9.3 Identify services run by web servers (e.g. scripting languages(client and server side scripting), databases, media)	● Basic knowledge of CSS, Javascript ● Front-end vs. back-end scripting
9.4 Identify performance issues. (e.g. bandwidth, internet connection types, pages loading slowly, resolution and size graphics.)	● Latency vs. bandwidth with internet connectivity ● Embedded graphics issues (JPG, PNG, GIF) with connection/speed ● Browser options
9.5 Differentiate among shared hosting, dedicated server and virtual private server(VPS).	● DSL/SDSL, VPS, VDSL, VPS ● Shared vs dedicated servers (pros and cons)
9.6 Identify Internet of Things(IOT) and common communication interfaces(e.g. Bluetooth, NFC, Wi-Fi, LTE)	● IoT in everyday life ● Types of iOT connections ● Security concerns of IoT ● Communication interfaces; ex. Bluetooth/Wi-Fi.
STANDARD 10.0 APPLY CLIENT-SIDE INTERNET SOFTWARE	
10.1 Identify key components and functions of Internet and web specialty browsers	● Internet ● Basic and speciality browses; ex. Chrome/ Safari/firefox/Maxthon

10.2 Use client collaboration sources/platforms(e.g. GitHub, Google Drive, DropBox, JSfiddle, browser developer tools)	● Basic collaboration platforms ● Safety concerns
10.3 Analyze remote computing tools and services and their application.	● VPNs ● Remote application tools; ex. Office suite/ google suite(docs/forms/sheets), Adobe Creative Suite
STANDARD 16.0 APPLY USER DESIGN PRINCIPLES TO INCLUDE WEBSITES AND APPLICATIONS	
16.1 Apply W3C standards and style conventions.	● W3C ● Standards and style conventions /Web development
16.2 Construct Web pages and applications that are compliant with ADA and sections 504 and 508 standards.	● create and develop web pages using W3C conventions ● ADA, Section 504 & 508 standards ● ADA, Section 504 & 508 requirements to web pages
16.3 Explain the concept of responsive design and applications.	● Optimal end user experience on web page through responsive design ● CSS to create responsive design for various browser/device compatibility ● Industry application tools available; ex. Bootstrap, Wirefy
16.4 Employ graphic methods to create images at specified locations.	● Method/functions to render/reposition an image
16.5 Choose correct GUI objects for input and output of data to the GUI Interface.	● Text boxes, labels, radio buttons, check buttons, dropdowns, list boxes in the GUI interface