Arizona CTE

Software and App Design

APPLY PROBLEM-SOLVING AND CRITICAL THINKING SKILLS SAD1.0

1. Establish objectives and outcomes for a task SAD1.1
2. Explain the process of decomposing a large programming problem into smaller, more manageable procedures SAD1.2
3. Explain "visualizing" as a problem-solving technique prior to writing code SAD1.3
4. Describe problem-solving and troubleshooting strategies applicable to software development SAD1.4

RECOGNIZE SECURITY ISSUES SAD2.0

1. Identify common computer threats (e.g., viruses, phishing, suspicious email, social engineering, spoofing, identity theft, and spamming) SAD2.1
2. Describe potential vulnerabilities in software (e.g., OWASP's Top 10) SAD2.2
3. Identify procedures to maintain data integrity and security (e.g., lock the screen, delete unrecognized emails, use trustworthy thumb drives, and use approved software) SAD2.3
4. Explain best practices to maintain integrity and security in software development (e.g., encryption, hashing, and digital signatures) SAD2.4
5. Describe methods for sanitizing user input to prevent issues (e.g., buffer overflows and SQL injection) SAD2.5
6. Explain the CIA (confidentiality, integrity, and availability) triad SAD2.6
7. Explain how software defects relate to software security (e.g., buffer overflows and cross-site scripting) SAD2.7

EXAMINE LEGAL AND ETHICAL ISSUES RELATED TO INFORMATION TECHNOLOGY SAD3.0

1. Explore intellectual property rights including software licensing and software duplication [e.g., Digital Millennium Copyright Act (DMCA), software licensing, and software duplication] SAD3.1
2. Compare and contrast open source and proprietary systems in relation to legal and ethical issues (e.g., data pricing, use of public and private networks, social networking, industry-related data, and data piracy) SAD3.2

	3 Identify issues and regulations affecting computers, other devices, the internet, and information privacy (e.g., HIPAA, COPPA, CISPA, FERPA, PCI, GDPR, and data brokers) SAD3.3
UTILIZE PRIMITIVE DATA TYPES AND STRINGS IN WRITING PROGRAMS SAD4.0	1 Declare numeric, Boolean, character, string variables, and float and double SAD4.1
	2 Choose the appropriate data type for a given situation SAD4.2
	3 Identify the correct syntax and usage for constants and variables in a program SAD4.3
	4 Identify the correct syntax and safe functions for operations on strings, including length, substring, and concatenation SAD4.4
	5 Explain complications of storing and manipulating data (i.e., the Big-O notation for analyzing storage and efficiency concerns, etc.) SAD4.5
	6 Research industry relevant programming languages (i.e., Java, JavaScript, Python, etc.) SAD4.6
PERFORM BASIC COMPUTER MATHEMATICS IN INFORMATION TECHNOLOGY SAD5.0	1 Apply basic mathematics to hardware (e.g., bits, bytes, kilobytes, megabytes, gigabytes, and terabytes) SAD5.1
	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 SAD5.2
	3 Identify and correctly use arithmetic operations applying the order of operations (precedence) with respect to programming SAD5.3
	4 Interpret and construct mathematical formulas SAD5.4
	5 Identify correct and problematic uses of integers, floating-point numbers, and fixed-point numbers in arithmetic SAD85.5
UTILIZE CONDITIONAL STRUCTURES IN WRITING PROGRAMS SAD6.0	1 Use the correct syntax for decision statements (e.g., if/else, if, and switch case) SAD6.1
	2 Compare values using relational operators (e.g., =, >, <, >=, <=, and not equal) SAD6.2
	3 Evaluate Boolean expressions (e.g., AND, OR, NOT, NOR, and XOR) SAD6.3
	4 Use the correct nesting for decision structures SAD6.4
UTILIZE ITERATIVE STRUCTURES IN	1 Identify various types of iteration structure (e.g., while, for, for-each, and recursion) SAD7.1

WRITING PROGRAMS SAD7.0	2 **Identify how loops are controlled (variable conditions and exits)** SAD7.2
	3 **Use the correct syntax for nested loops** SAD7.3
	4 **Compute the values of variables involved with nested loops** SAD7.4
UTILIZE BASIC DATA STRUCTURES IN WRITING PROGRAMS SAD8.0	1 **Demonstrate basic uses of arrays including initialization, storage, and retrieval of values** SAD8.1
	2 **Distinguish between arrays and hash maps (associative arrays)** SAD8.2
	3 **Identify techniques for declaring, initializing, and modifying user-defined data types** SAD8.3
	4 **Search and sort data in an array** SAD8.4
	5 **Create and use two-dimensional arrays** SAD8.5
	6 **Describe the efficiency of different sorting algorithms (e.g., bubble, insertion, and merge)** SAD8.6
	7 **Describe the efficiency of linear vs. binary searches [e.g., O(n) and O(log n)]** SAD8.7
IDENTIFY INTERNET PROTOCOLS AND OPERATIONS SAD9.0	1 **Explain cloud-based computing and content delivery networks** SAD9.1
	2 **Identify the components and functions of the internet (e.g., HTTP, HTTPS, FTP, IP addresses, and IMAP)** SAD9.2
	3 **Identify services run by web servers [e.g., scripting languages (client- and server-side scripting), databases, and media]** SAD9.3
	4 **Identify performance issues (e.g., bandwidth, internet connection types, pages loading slowly, resolution, and size graphics)** SAD9.4
	5 **Differentiate among shared hosting, dedicated server, and virtual private server (VPS)** SAD9.5
	6 **Identify Internet of Things (IOT) and common communication interfaces (e.g., Bluetooth, NFC, Wi-Fi, and LTE)** SAD9.6
APPLY CLIENT-SIDE INTERNET SOFTWARE SAD10.0	1 **Identify key components and functions of internet and web specialty browsers** SAD10.1
	2 **Use client collaboration sources/platforms (e.g., GitHub, Google Drive, Dropbox, JSFiddle, and browser developer tools)** SAD10.2
	3 **Analyze remote computing tools and services and their application** SAD10.3

DEMONSTRATE PROGRAM ANALYSIS AND DESIGN SAD11.0

1. Implement the steps in the System Development Life Cycle (SDLC) (e.g., planning, analysis, design, development, testing, implementation, and maintenance) SAD11.1
2. Develop program requirements/specifications and a testing plan (e.g., user stories, automated testing, and test procedures) SAD11.2
3. Apply pseudocode or graphical representations to plan the structure of a program or module (e.g., flowcharting, white boarding, and UML) SAD11.3
4. Create and implement basic algorithms SAD11.4

DEVELOP A PROGRAM SAD12.0

1. Use a program editor to enter and modify code SAD12.1
2. Identify correct input/output statements SAD12.2
3. Choose the correct method of assigning input to variables including data sanitization SAD12.3
4. Choose the correct method of outputting data with formatting and escaping SAD12.4
5. Differentiate between interpreted and compiled code (e.g., steps necessary to run executable code) SAD12.5
6. Identify the purpose of a build system (e.g., make, rake, ant, maven, SCons, and grunt) SAD12.6
7. Apply industry standards in documentation (e.g., self-documenting code; function-level, program-level, and user-level documentation) SAD12.7
8. Name identifiers and formatting code by applying recognized conventions SAD12.8
9. Demonstrate refactoring techniques to reduce repetitious code and improve maintainability SAD12.9
10. Demonstrate the use of parameters to pass data into program modules SAD12.10
11. Demonstrate the use of return values from modules SAD12.11

TEST AND DEBUG TO VERIFY PROGRAM OPERATION SAD13.0

1. Identify errors in program modules SAD13.1
2. Identify boundary cases and generate appropriate test data SAD13.2
3. Perform integration testing including tests within a program to protect execution from bad input or other run-time errors SAD13.3
4. Categorize, identify, and correct errors in code, including syntax, semantic, logic, and runtime SAD13.4

	5 Perform different methods of debugging (e.g., hand-trace code and real time debugging tools) SAD13.5
UTILIZE AND CREATE COMMUNITY RESOURCES SAD14.0	1 Use standard library functions SAD14.1
	2 Find and use third party libraries (e.g., web-based and package managers) SAD14.2
	3 Explain and interact with an Application Program Interface (API) SAD14.3
USE VERSION CONTROL SYSTEMS SAD15.0	1 Identify the purpose of version control systems (e.g., Git and Mercurial) SAD15.1
	2 Create a new repository SAD15.2
	3 Add, push, and pull source code from repository SAD15.3
	4 Explain branching and its uses SAD15.4
	5 Restore previous versions of code from the repository SAD15.5
APPLY USER DESIGN PRINCIPLES TO INCLUDE WEBSITES AND APPLICATIONS SAD16.0	1 Apply W3C standards and style conventions SAD16.1
	2 Construct web pages and applications that are compliant with ADA and sections 504 and 508 standards SAD16.2
	3 Explain the concept of responsive design and applications SAD16.3
	4 Employ graphics methods to create images at specified locations SAD16.4
	5 Choose correct GUI objects for input and output of data to the GUI interface (e.g., text boxes, labels, radio buttons, check boxes, dropdowns, and list boxes) SAD16.5.
USE AND UPDATE DATA STORAGE AND MANAGEMENT SAD17.0	1 Input/output data from a sequential file or database SAD17.1
	. Demonstrate creating, reading, updating, and dropping a database SAD17.2
	3 Demonstrate the proper use of SQL database applications that work with different languages (e.g., MongoDB, Microsoft Access, Oracle Databases, and Code.org's App Lab) SAD17.3
EMPLOY OBJECT-ORIENTED PROGRAMMING TECHNIQUES SAD18.0	1 Make a distinction between an object and a class SAD18.1
	2 Differentiate among inheritance, composition, and class relationships SAD18.2
	3 Instantiate objects from existing classes SAD18.3
	4 Read the state of an object by invoking accessor methods SAD18.4

	5 **Change the state of an object by invoking a modifier method** SAD18.5
	6 **Determine the requirements for constructing new objects by reading the documentation** SAD18.6
	7 **Create a user-defined class** SAD18.7
	8 **Create a subclass of an existing class** SAD18.8
	9 **Identify the use of an abstract class as opposed to an interface** SAD18.9
	10 **Explain the object-oriented concepts of polymorphism, inheritance, and encapsulation** SAD18.10
EMPLOY RUNTIME AND ERROR HANDLING TECHNIQUES SAD19.0	1 **Identify runtime errors** SAD19.1
	2 **Describe error handling strategies** SAD19.2
	3 **Handle unexpected return values** SAD19.3
	4 **Handle (catch) runtime errors and take appropriate action** SAD19.4
	5 **Throw standard exception classes** SAD19.5
	6 **Develop and throw custom exception classes** SAD19.6