Computer Science
K–12

Section 05

Computer Science K–12

1 Knowledge of computational thinking and problem solving

1. Analyze a problem and apply appropriate solution strategies.
2. Apply the steps of algorithmic problem solving when designing solutions to problems.
3. Apply the stages of the software development life cycle (i.e., problem definition, analysis, design, testing, implementation, maintenance).
4. Determine and select an appropriate algorithm for a given problem.
5. Predict outputs of algorithms for a given input.
6. Identify an appropriate set of data necessary for testing a computer solution.

2 Knowledge of data types and structures

1. Distinguish between constants and variables and between local and global identifiers.
2. Distinguish between integer, real number, character, string, Boolean, and object data types.
3. Recognize and convert between binary, decimal, and hexadecimal number systems.
4. Identify characteristics and uses of data structures, including arrays, linked lists, stacks, queues, and sets.
5. Distinguish between instance, class, and local variables in an object-oriented design.
6. Identify components of class declarations for an object-oriented program and distinguish between public and private access specifiers.

3 Knowledge of programming logic

1. Distinguish between error types (e.g., syntax, runtime, logic) and apply principles of debugging.
2. Identify principles, characteristics, and uses of internal and external program documentation.
3. Analyze the characteristics and functions of object-oriented and procedural languages.
4. Select the appropriate algorithmic sequence, conditional, iteration, and recursive constructs for a given purpose.
5. Analyze characteristics and applications of searching (i.e., sequential, binary) and sorting (i.e., selection, insertion, merge) algorithms.

6. Analyze the characteristics and applications of propositional logic (e.g., De Morgan's laws).

4 Knowledge of programming languages

1. Identify characteristics and apply concepts of the Scratch™ programming language learning environment from the MIT Media Library.

2. Analyze segments of Java® code containing sequential, conditional, or iteration statements.

3. Analyze segments of Java® code involving methods, interacting objects, or passing parameters.

4. Apply principles of data types and data manipulation (e.g., string methods, arithmetic operations) in the Java® programming language.

5. Apply principles of abstraction, encapsulation, inheritance, and polymorphism in the Java® programming language.

5 Knowledge of computer hardware, software, and networking

1. Identify the hardware components of a computer system and their functions (e.g., input, output, processing, storage).

2. Analyze the advantages, disadvantages, or both of various data storage technologies.

3. Identify the characteristics and uses of various types of software (e.g., system, application).

4. Apply features and functions of application and productivity software (e.g., word processing, spreadsheet, database, multimedia authoring, Web development software).

5. Identify concepts and terminology related to networks (e.g., network protocols, Open Systems Interconnection model, client-server, cloud computing).

6. Identify characteristics and uses of network devices (e.g., servers, routers, switches, access points, workstations).

6 Knowledge of the historical aspects and social issues related to computer technologies

1. Identify examples of appropriate use (e.g., software license types, archival copying, fair use of copyrighted materials) and misuse (e.g., plagiarism, music and video piracy) of intellectual property.

2. Identify milestones in the historical development of computer technology and important contributions of individuals or groups to the development of computer technology.

---

3. Analyze cultural, legal, and ethical issues and responsibilities of digital citizens, organizations, and government entities (e.g., privacy issues related to Internet use, data protection).

4. Analyze issues related to malicious software, social engineering, and security awareness.

5. Identify concepts and terminology related to security countermeasures (e.g., firewalls, antivirus programs, filtering software, encryption) that prevent, detect, and correct breaches.

6. Analyze security issues related to maintaining the confidentiality, integrity, and availability of information.

7 Knowledge of computer science pedagogy

1. Apply appropriate and effective classroom management strategies for teaching computer science (e.g., laboratory work, cooperative learning, electronic communications).

2. Apply appropriate and effective instructional strategies for teaching computer science (e.g., independent learning, case studies, role-playing, manipulatives, visualizations, simulations, modeling, team software development).

3. Apply appropriate and effective formative and summative assessment strategies for teaching computer science (e.g., rubrics, portfolios).

4. Apply appropriate and effective accommodations, adaptations, and strategies that ensure the equitable use of technology for diverse student populations (e.g., students with exceptionalities, English language learners, students from various socioeconomic levels).

5. Determine characteristics and apply uses of instructional technologies (e.g., collaborative online tools, social networking, computer-based learning, mobile devices).

6. Recognize opportunities, skills, and paths related to college and career readiness in the field of computer science.

7. Apply practices for planning and developing curricula that meet state and national standards and recognize resources for ongoing professional support and development.