| **Test Content Categories** | **How well do I know the content?  (scale 1–5)** | **What resources do I have/need for this content?** | **Where can I find the resources I need?** | **Dates I will study this content** | **Date completed** |
| --- | --- | --- | --- | --- | --- |
| **I. Impacts of Computing (15%)** |  |  |  |  |  |
| **A. Understands and applies knowledge of impact of, obstacles to, and effects of computing** |  |  |  |  |  |
| 1. Understand computing as a way of expressing creativity, solving problems, enabling communication, and fostering innovation in a variety of fields and careers |  |  |  |  |  |
| a. recognize that computers can be used to showcase creativity |  |  |  |  |  |
| b. recognize the benefits of using computers to solve problems |  |  |  |  |  |
| c. provide examples of how computers enable communication and collaboration |  |  |  |  |  |
| d. provide examples of how computers foster innovation |  |  |  |  |  |
| 2. Know the obstacles to equal access to computing among different groups and the impact of those obstacles |  |  |  |  |  |
| a. identify obstacles to equal access to computing among different groups (e.g., groups defined by gender, socioeconomic status, disability/accessibility needs) and the impact of those obstacles |  |  |  |  |  |
| b. identify factors that contribute to the digital divide |  |  |  |  |  |
| c. match obstacles to equal access with effective solutions |  |  |  |  |  |
| 3. Understand beneficial and harmful effects of computing innovations and the trade-offs between them |  |  |  |  |  |
| a. analyze computing innovations in terms of their social, economic, and cultural impacts, both beneficial and harmful |  |  |  |  |  |
| b. identify trade-offs between beneficial and harmful effects of computer innovations |  |  |  |  |  |
| **B. Understands and applies knowledge of issues regarding intellectual property, ethics, privacy, and security in computing** |  |  |  |  |  |
| 1. Know different methods of protecting intellectual property rights and the trade-offs between them in a variety of contexts (e.g., Creative Commons, open source, copyright) |  |  |  |  |  |
| a. using correct vocabulary, describe how different methods of protecting intellectual property rights work |  |  |  |  |  |
| b. given a context, identify appropriate methods of protecting intellectual property rights |  |  |  |  |  |
| c. identify and compare trade-offs between different methods of protecting intellectual property rights |  |  |  |  |  |
| 2. Understand ethical and unethical computing practices and their social, economic, and cultural implications |  |  |  |  |  |
| a. identify ethical and unethical computing practices in context |  |  |  |  |  |
| b. describe the social, economic, and cultural implications of ethical and unethical computing practices |  |  |  |  |  |
| c. identify the conditions under which a given computing practice is ethical or legal |  |  |  |  |  |
| 3. Know privacy and security issues regarding the acquisition, use, and disclosure of information in a digital world |  |  |  |  |  |
| a. using correct vocabulary, describe privacy and security issues |  |  |  |  |  |
| b. in context, identify appropriate strategies to safeguard privacy and ensure security |  |  |  |  |  |
| c. describe trade-offs between local and cloud-based data storage |  |  |  |  |  |
| d. identify methods that digital services use to collect information about users |  |  |  |  |  |
| **II. Algorithms and Computational Thinking (25%)** |  |  |  |  |  |
| **A. Understands and applies knowledge of abstraction, pattern recognition, problem decomposition, number base conversion, and algorithm formats** |  |  |  |  |  |
| 1. Understand abstraction as a foundation of computer science |  |  |  |  |  |
| a. identify, create, or complete the correct ordering, from low to high, of an abstraction hierarchy |  |  |  |  |  |
| b. identify abstractions in context |  |  |  |  |  |
| c. identify details that can be removed from a solution in order to generalize it |  |  |  |  |  |
| 2. Know how to use pattern recognition, problem decomposition, and abstraction to develop an algorithm |  |  |  |  |  |
| a. given a table of values or other data source, identify the patterns in the data and identify algorithms that could produce the patterns |  |  |  |  |  |
| b. identify components that could be part of an algorithm to solve a problem |  |  |  |  |  |
| c. identify actions and actors when decomposing a problem |  |  |  |  |  |
| d. identify appropriate decomposition strategies |  |  |  |  |  |
| 3. Understand number base conversion and binary, decimal, and hexadecimal number systems |  |  |  |  |  |
| a. convert between number bases |  |  |  |  |  |
| b. analyze and compare representations of numbers in different bases |  |  |  |  |  |
| 4. Understand how to develop and analyze algorithms expressed in multiple formats (e.g., natural language, flowcharts, pseudocode) |  |  |  |  |  |
| a. interpret diagrams that describe algorithms, given an explanation of the symbols used |  |  |  |  |  |
| b. compare algorithms written in multiple formats |  |  |  |  |  |
| c. trace and analyze algorithms written in different formats |  |  |  |  |  |
| d. identify correct sequencing of steps in an algorithm and errors in sequencing |  |  |  |  |  |
| **B. Understands and applies knowledge of algorithm analysis, searching and sorting algorithms, recursive algorithms, and randomization** |  |  |  |  |  |
| 1. Be familiar with the limitations of computing in terms of time, space, and solvability as well as with the use of heuristic solutions that can address these limitations |  |  |  |  |  |
| a. identify and compare algorithms that are linear, quadratic, exponential, or logarithmic |  |  |  |  |  |
| b. recognize the existence of problems that cannot be solved by a computer |  |  |  |  |  |
| c. in context, identify factors that prevent a problem from being solvable |  |  |  |  |  |
| d. identify situations where heuristic solutions are useful |  |  |  |  |  |
| e. in context, identify space and time limitations of computational solutions to problems |  |  |  |  |  |
| 2. Understand searching and sorting algorithms; can analyze sorting algorithms for correctness and can analyze searching algorithms for correctness and efficiency |  |  |  |  |  |
| a. trace algorithms and predict output and intermediate results |  |  |  |  |  |
| b. calculate the number of comparisons required for linear and binary search algorithms |  |  |  |  |  |
| 3. Understand simple recursive algorithms (e.g., n factorial, sum of first n integers) |  |  |  |  |  |
| a. trace simple recursive algorithms |  |  |  |  |  |
| b. provide missing steps in incomplete simple recursive algorithms |  |  |  |  |  |
| c. identify parts of a recursive algorithm (e.g., base or stopping condition, recursive call) |  |  |  |  |  |
| d. identify errors in simple recursive algorithms |  |  |  |  |  |
| e. identify an iterative algorithm that is equivalent to a recursive algorithm |  |  |  |  |  |
| 4. Be familiar with the use of randomization in computing |  |  |  |  |  |
| a. identify appropriate uses of randomization in a variety of applications |  |  |  |  |  |
| b. identify the difference between random and pseudorandom numbers |  |  |  |  |  |
| **III. Programming (30%)** |  |  |  |  |  |
| **A. Understands and applies knowledge of programming control structures, standard operators, variables, correctness, extensibility, modifiability, and reusability** |  |  |  |  |  |
| 1. Understand how to write and modify computer programs in a text-based programming language |  |  |  |  |  |
| a. describe what a program does or be able to choose the code segment that correctly implements a given intended purpose |  |  |  |  |  |
| b. identify missing code in a code segment with a stated intended purpose |  |  |  |  |  |
| c. place statements in appropriate order to create a correct program |  |  |  |  |  |
| d. identify how changing one part of a code segment will affect the output |  |  |  |  |  |
| 2. Understand how to analyze computer programs in terms of correctness |  |  |  |  |  |
| a. trace code and indicate the output printed or the value of variables after code segment execution |  |  |  |  |  |
| b. indicate the inputs that produce given outputs for a code segment |  |  |  |  |  |
| c. describe what a program does or choose the code segment that correctly implements a given intended purpose |  |  |  |  |  |
| d. identify valid preconditions and postconditions |  |  |  |  |  |
| e. compare two code segments or algorithms |  |  |  |  |  |
| f. identify the type of error produced by a code segment (i.e., syntax, runtime, compile-time, overflow, round-off, logic) |  |  |  |  |  |
| g. identify errors in incorrect code and changes that can be made to correct them |  |  |  |  |  |
| 3. Know the concepts of extensibility, modifiability, and reusability |  |  |  |  |  |
| a. identify the meaning of the terms |  |  |  |  |  |
| b. identify functionally equivalent statements or code segments that differ in one of these three ways |  |  |  |  |  |
| c. identify situations where the use of constants or variables would be preferred over hard-coded values |  |  |  |  |  |
| d. identify opportunities for parameterization |  |  |  |  |  |
| e. choose code that improves on given code by making it more extensible, modifiable, or reusable |  |  |  |  |  |
| f. identify changes that would improve a given code segment |  |  |  |  |  |
| 4. Understand the three basic constructs used in programming: sequence, selection, and iteration |  |  |  |  |  |
| a. trace code and indicate the output printed or the value of variables after code segment execution |  |  |  |  |  |
| b. indicate inputs that produce given outputs for a code segment |  |  |  |  |  |
| c. describe what a program does or choose the code segment that correctly implements a given intended purpose |  |  |  |  |  |
| d. identify missing code in a code segment with a stated intended purpose |  |  |  |  |  |
| e. identify equivalent statements or code segments |  |  |  |  |  |
| f. identify the three constructs when used in code |  |  |  |  |  |
| g. identify which of the constructs are needed to implement given functionality |  |  |  |  |  |
| h. convert code that does not use iteration to equivalent code that uses iteration |  |  |  |  |  |
| 5. Understand how to use standard operators (i.e., assignment, arithmetic, relational, logical) and operator precedence to write programs |  |  |  |  |  |
| a. trace code and indicate the output displayed or the value of variables after code segment execution |  |  |  |  |  |
| b. indicate inputs that produce given outputs for a code segment |  |  |  |  |  |
| c. describe what a program does or choose the code segment that correctly implements a stated intended purpose |  |  |  |  |  |
| d. identify missing code in a code segment with a stated intended purpose |  |  |  |  |  |
| e. identify equivalent statements or code segments |  |  |  |  |  |
| f. place statements in appropriate order to create a correct program |  |  |  |  |  |
| g. use Boolean algebra to identify equivalent Boolean expressions |  |  |  |  |  |
| h. write a Boolean expression equivalent to given code, or identify code equivalent to a given Boolean expression or English description |  |  |  |  |  |
| i. identify the correct implementation of a given formula, including formulas with fractions |  |  |  |  |  |
| j. evaluate expressions that include arithmetic operations |  |  |  |  |  |
| 6. Understand how to use variables and a variety of data types |  |  |  |  |  |
| a. identify variables and data types (e.g., integers, floating point, string, Booleans, arrays/lists) |  |  |  |  |  |
| b. identify the need for type conversion |  |  |  |  |  |
| c. trace code and indicate the output printed or the value of variables after code segment execution |  |  |  |  |  |
| d. indicate the inputs that produce given outputs for a code segment |  |  |  |  |  |
| e. describe what a program does or choose the code segment that correctly implements a stated intended purpose |  |  |  |  |  |
| f. identify missing code in a code segment with a stated intended purpose |  |  |  |  |  |
| g. identify equivalent statements or code segments |  |  |  |  |  |
| h. place statements in appropriate order to create a correct program |  |  |  |  |  |
| i. describe the difference between integer and floating point numeric data types |  |  |  |  |  |
| j. describe the difference between integer and floating point division |  |  |  |  |  |
| k. describe the benefits of the use of each data type |  |  |  |  |  |
| l. distinguish between global and local scope |  |  |  |  |  |
| m. identify the most appropriate data type in a given context |  |  |  |  |  |
| n. identify the correct sequence of string operations to produce a given output |  |  |  |  |  |
| **B. Understands and applies knowledge of procedures, event-driven programs, usability, data structures, debugging, documenting and reviewing code, libraries and A**​**P**​**Is, I**​**D**​**Es, and programming language paradigms, including object-oriented concepts** |  |  |  |  |  |
| 1. Understand how to write and call procedures with parameters and return values |  |  |  |  |  |
| a. trace code and indicate the output printed or the value of variables after code segment execution |  |  |  |  |  |
| b. indicate inputs that produce given outputs for a code segment |  |  |  |  |  |
| c. describe what a program does or choose the code segment that correctly implements a stated intended purpose |  |  |  |  |  |
| d. identify missing code in a code segment with a stated intended purpose |  |  |  |  |  |
| e. identify equivalent statements or code segments |  |  |  |  |  |
| f. place statements in appropriate order to create a correct program |  |  |  |  |  |
| g. trace code when references to objects and arrays are passed to procedures |  |  |  |  |  |
| h. trace code that includes nested procedure calls |  |  |  |  |  |
| 2. Know the concepts of event-driven programs that respond to external events (e.g., sensors, messages, clicks) |  |  |  |  |  |
| a. trace code and indicate the output printed or the value of variables after code segment execution |  |  |  |  |  |
| b. indicate inputs that produce given outputs for a code segment |  |  |  |  |  |
| c. describe what a program does or choose the code segment that correctly implements a stated intended purpose |  |  |  |  |  |
| d. identify missing code in a code segment with a stated intended purpose |  |  |  |  |  |
| e. identify possible errors due to asynchronous events |  |  |  |  |  |
| f. identify aspects of concurrency in event-driven programming |  |  |  |  |  |
| 3. Be familiar with usability and user experience (e.g., ease of use and accessibility) |  |  |  |  |  |
| a. identify code that improves on given code in terms of usability or user experience |  |  |  |  |  |
| b. identify meaningful error messages |  |  |  |  |  |
| c. identify features that improve accessibility |  |  |  |  |  |
| 4. Be familiar with dictionaries/maps, stacks, and queues |  |  |  |  |  |
| a. identify a data structure based on a description of behavior or appropriate use |  |  |  |  |  |
| b. given goals, constraints, or context, identify the most appropriate data structure |  |  |  |  |  |
| c. trace code that uses a particular data structure |  |  |  |  |  |
| 5. Understand how to use debugging techniques and appropriate test cases |  |  |  |  |  |
| a. identify which test cases are most useful for given code |  |  |  |  |  |
| b. differentiate between different types of errors (e.g., overflow, round-off, syntax, runtime, compile-time, logic) |  |  |  |  |  |
| c. describe useful debugging techniques (e.g., where to put print statements) |  |  |  |  |  |
| d. differentiate between empirical testing and proof |  |  |  |  |  |
| e. identify errors in code and solutions to those errors |  |  |  |  |  |
| 6. Be familiar with characteristics of well-documented computer programs that are usable, readable, and modular |  |  |  |  |  |
| a. identify characteristics of good documentation |  |  |  |  |  |
| b. identify good and poor documentation practices in context |  |  |  |  |  |
| 7. Be familiar with techniques to obtain and use feedback to produce high-quality code (e.g., code reviews, peer feedback, end user feedback) |  |  |  |  |  |
| a. identify situations in which each of the three listed techniques are useful |  |  |  |  |  |
| 8. Know how to use libraries and A​P​Is |  |  |  |  |  |
| a. identify correct call(s) and use of return values given an A​P​I definition |  |  |  |  |  |
| b. identify reasons to use or not use libraries in place of writing original code |  |  |  |  |  |
| c. identify applications (e.g., math libraries, random number generation) that use AP Is |  |  |  |  |  |
| 9. Understand programming techniques to validate correct input and detect incorrect input |  |  |  |  |  |
| a. identify effective input data validation strategies |  |  |  |  |  |
| b. compare data validation (proper range and format) and data verification (e.g., password verification) |  |  |  |  |  |
| c. identify improvements to code for which data validation is required |  |  |  |  |  |
| 10. Be familiar with the features and capabilities of integrated development environments (I​D​Es) |  |  |  |  |  |
| a. identify components of I​D​Es |  |  |  |  |  |
| b. identify benefits and drawbacks of using I​D​Es |  |  |  |  |  |
| c. identify the costs and benefits of context editors |  |  |  |  |  |
| 11. Be familiar with the differences between low- and high-level programming languages |  |  |  |  |  |
| a. identify characteristics of low- and high-level languages |  |  |  |  |  |
| 12. Be familiar with different programming paradigms |  |  |  |  |  |
| a. identify the terminology of procedural programming |  |  |  |  |  |
| b. identify the terminology of object-oriented programming |  |  |  |  |  |
| c. compare programming paradigms |  |  |  |  |  |
| 13. Know object-oriented programming concepts |  |  |  |  |  |
| a. identify classes, instance variables, and methods given a diagram |  |  |  |  |  |
| b. identify the benefits of inheritance and encapsulation |  |  |  |  |  |
| c. identify distinctions between overloading and overriding |  |  |  |  |  |
| 14. Be familiar with program compilation and program interpretation |  |  |  |  |  |
| a. identify differences between compilation and interpretation |  |  |  |  |  |
| b. identify differences between source code and object code |  |  |  |  |  |
| **IV. Data (15%)** |  |  |  |  |  |
| A. Understands and applies knowledge of digitalization, data encryption and decryption, and computational tools |  |  |  |  |  |
| 1. Understand bits as the universal medium for expressing digital information |  |  |  |  |  |
| a. perform calculations, using bits and bytes |  |  |  |  |  |
| b. determine the number of bits and bytes required to store a given amount of data |  |  |  |  |  |
| c. given the description of an encoding scheme, encode or decode data |  |  |  |  |  |
| d. describe lossy and lossless data compression |  |  |  |  |  |
| e. explain why binary numbers are fundamental to the operation of computer systems |  |  |  |  |  |
| 2. Be familiar with concepts of data encryption and decryption |  |  |  |  |  |
| a. distinguish between encoding and encryption |  |  |  |  |  |
| b. identify trade-offs in the use of data encryption |  |  |  |  |  |
| 3. Know how to use computational tools, including spreadsheets, to analyze data in order to discover, explain, and visualize patterns, connections, and trends |  |  |  |  |  |
| a. transform data to make it more useful |  |  |  |  |  |
| b. identify specific data or characteristics of specific data that need to be removed or modified before an entire data set can be used |  |  |  |  |  |
| c. describe the use of spreadsheet operations (e.g., formulas, filters, sorts, charts, graphs) to analyze and visualize data |  |  |  |  |  |
| **B. Understands and applies knowledge of simulation, modeling, and manipulation of data** |  |  |  |  |  |
| 1. Be familiar with the use of computing in simulation and modeling |  |  |  |  |  |
| a. describe questions that can be answered with a given simulation, or explain what data and process are required in a simulation in order to answer a given question |  |  |  |  |  |
| b. trace code in a simulation context |  |  |  |  |  |
| c. identify missing code in a simulation context |  |  |  |  |  |
| d. identify the impact of changes to simulations (e.g., more or fewer variables, more or less data) |  |  |  |  |  |
| e. identify applications of simulation and modeling |  |  |  |  |  |
| 2. Be familiar with methods to store, manage, and manipulate data |  |  |  |  |  |
| a. use terminology and concepts of files and databases |  |  |  |  |  |
| b. identify measures of file size (e.g., byte, kilo, mega, giga, tera, peta) |  |  |  |  |  |
| c. identify issues connected with the storage requirements of computing applications, including scale, redundancy, and backup |  |  |  |  |  |
| 3. Be familiar with a variety of computational methods for data collection, aggregation, and generation |  |  |  |  |  |
| a. identify the benefits of working with publicly available data sets |  |  |  |  |  |
| b. identify the types of data generated by surveys and sensors |  |  |  |  |  |
| c. identify examples of crowdsourcing and citizen science |  |  |  |  |  |
| d. identify appropriate data-collection methods for a given context and purpose |  |  |  |  |  |
| **V. Computing Systems and Networks (15%)** |  |  |  |  |  |
| **A. Understands and applies knowledge of operating systems, computing systems, communication between devices, and cloud computing** |  |  |  |  |  |
| 1. Know that operating systems are programs that control and coordinate interactions between hardware and software components |  |  |  |  |  |
| a. identify hardware components and their functions |  |  |  |  |  |
| b. identify software components and their functions |  |  |  |  |  |
| c. identify common operating systems tasks |  |  |  |  |  |
| d. identify resource issues that have an impact on functionality |  |  |  |  |  |
| 2. Be familiar with computing systems embedded in everyday objects (e.g., Internet of Things [I​o​T], A​T​Ms, medical devices) |  |  |  |  |  |
| a. describe what an embedded system is |  |  |  |  |  |
| b. define what the I​o​T is and how it is used |  |  |  |  |  |
| c. describe how sensors are used in embedded systems |  |  |  |  |  |
| 3. Know the capabilities, features, and uses of different types of computing systems (e.g., desktop, mobile, cluster) |  |  |  |  |  |
| a. identify capabilities, features, and uses for each type of computer system |  |  |  |  |  |
| b. identify criteria to evaluate and compare computing systems |  |  |  |  |  |
| 4. Be familiar with computers as layers of abstraction from hardware (e.g., logic gates, chips) to software (e.g., system software, applications) |  |  |  |  |  |
| a. identify appropriate abstraction layers for hardware and software components |  |  |  |  |  |
| 5. Be familiar with the steps required to execute a computer program (fetch-decode-execute cycles) |  |  |  |  |  |
| a. describe what happens during fetch, decode, and execute, including the order of the steps in the cycle |  |  |  |  |  |
| 6. Be familiar with trade-offs between local, network, and cloud computing and storage |  |  |  |  |  |
| a. identify advantages and disadvantages in terms of performance, cost, security, reliability, and collaboration |  |  |  |  |  |
| b. identify means of storing binary data |  |  |  |  |  |
| 7. Be familiar with communication between devices |  |  |  |  |  |
| a. identify and compare wireless communication systems |  |  |  |  |  |
| b. identify and compare wired communication systems |  |  |  |  |  |
| c. identify and compare network types |  |  |  |  |  |
| **B. Understands and applies knowledge of networks, including security issues and the Web** |  |  |  |  |  |
| 1. Know components of networks |  |  |  |  |  |
| a. identify network hardware devices and their functions |  |  |  |  |  |
| b. describe possible abstraction models of networks |  |  |  |  |  |
| 2. Be familiar with factors that have an impact on network functionality |  |  |  |  |  |
| a. define basic terminology (e.g., bandwidth, load, latency) |  |  |  |  |  |
| b. estimate necessary bandwidth and data size for a given situation |  |  |  |  |  |
| c. identify critical resources for a given situation |  |  |  |  |  |
| 3. Be familiar with how Internet and Web protocols work |  |  |  |  |  |
| a. describe the purpose of protocols and identify common Internet and Web protocols |  |  |  |  |  |
| b. compare I​P​v​4 and I​P​v​6 |  |  |  |  |  |
| c. identify and describe the basic parts of a U​R​L (e.g., protocol, subdomain, domain name, port, path) |  |  |  |  |  |
| d. describe the hierarchical structure of names in the domain name system (D​N​S) |  |  |  |  |  |
| e. describe the purpose and function of I​P addressing |  |  |  |  |  |
| f. identify how Internet protocols address reliability, redundancy, and error handling |  |  |  |  |  |
| 4. Be familiar with digital and physical strategies for maintaining security |  |  |  |  |  |
| a. identify characteristics of strong passwords (e.g., length, bits per character) |  |  |  |  |  |
| b. identify digital and physical security strategies |  |  |  |  |  |
| c. identify trade-offs in the use of security measures (e.g., encryption, decryption, digital signatures and certificates) |  |  |  |  |  |
| 5. Be familiar with concepts of cybersecurity |  |  |  |  |  |
| a. identify and define the five pillars of cybersecurity: confidentiality, integrity, availability, nonrepudiation, and authentication |  |  |  |  |  |
| 6. Be familiar with the components that make up the Web (e.g., H​T​T​P, H​T​M​L, browsers, servers, clients) |  |  |  |  |  |
| a. identify the uses of markup languages |  |  |  |  |  |
| b. identify the purposes of browsers, servers, and clients |  |  |  |  |  |