AP Computer Science Principles

Concept Outline

Big Idea 1: Creative Development

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2020 | | Rationale for changes | 2018 | |
| Learning Objectives | Essential Knowledge |  | Learning Objectives | Essential Knowledge |
| Students must be able to… | Students will know that… |  | Students must be able to… | Students will know that… |
|  | REMOVED |  | 1.2.1 Create a computational artifact for creative expression. [P2] | 1.2.1A A computational artifact is anything created by a human using a computer and can be, but is not limited to, a program, an image, audio, video, a presentation, or a web page file.  1.2.1B Creating computational artifacts requires understanding and using software tools and services.  1.2.1C Computing tools and techniques are used to create computational artifacts and can include, but are not limited to, programming IDEs, spreadsheets, 3D printers, or text editors.  1.2.1D A creatively developed computational artifact can be created by using nontraditional, nonprescribed computing techniques.  1.2.1E Creative expressions in a computational artifact can reflect personal expressions of ideas or interests. |
|  | REMOVED |  | 1.2.2 Create a computational artifact using computing tools and techniques to solve a problem. [P2] | 1.2.2A Computing tools and techniques can enhance the process of finding a solution to a problem.  1.2.2B A creative development process for creating computational artifacts can be used to solve problems when traditional or prescribed computing techniques are not effective. |
|  | REMOVED |  | 1.3.1 Use computing tools and techniques for creative expression. [P2] | 1.3.1A Creating digital effects, images, audio, video, and animations has transformed industries.  1.3.1B Digital audio and music can be created by synthesizing sounds, sampling existing audio and music, and recording and manipulating sounds, including layering and looping.  1.3.1C Digital images can be created by generating pixel patterns, manipulating existing digital images, or combining images.  1.3.1D Digital effects and animations can be created by using existing software or modified software that includes functionality to implement the effects and animations.  1.3.1E Computing enables creative exploration of both real and virtual phenomena. |
| Explain how computing innovations are improved through collaboration. | * A computing innovation ~~is a program or an innovation that~~ includes a program as an integral part of its function. * A computing innovation can be physical (i.e., self-driving car), non-physical computing software (i.e., picture editing software), or non-physical computing concepts (i.e., eCommerce). * Effective collaboration produces a computing innovation that reflects the diversity of talents and perspectives of those who designed it. * Collaboration that includes diverse perspectives helps to avoid bias in the development of computing innovations. * Consultation and communication with users is an important aspect of the development of computing innovations. * Research gathered from users can be used to understand the purpose of a program from diverse perspectives in order to develop a program that can be used to fully incorporate these perspectives. |  | 1.2.4 Collaborate in the creation of computational artifacts. [P6]  5.1.3 Collaborate to develop a program. [P6] | 1.2.4A A collaboratively created computational artifact reflects effort by more than one person.  1.2.4B Effective collaborative teams consider the use of online collaborative tools.  1.2.4C Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation.  1.2.4D Effective collaboration strategies enhance performance.  1.2.4E Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts.  1.2.4F A collaboratively created computational artifact can reflect personal expressions of ideas.  5.1.3A Collaboration can decrease the size and complexity of tasks required of individual programmers.  5.1.3B Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming.  5.1.3C Collaboration in the iterative development of a program requires different skills than developing a program alone.  5.1.3D Collaboration can make it easier to find and correct errors when developing programs.  5.1.3E Collaboration facilitates developing program components independently.  5.1.3F Effective communication between participants is required for successful collaboration when developing programs. |
| Explain how computing innovations are developed by groups of people. | Online tools support collaboration by allowing programmers to virtually share and provide feedback on ideas and documents.  Common models such as pair programming exist to facilitate collaboration. |  | 1.2.4 Collaborate in the creation of computational artifacts. [P6]  5.1.3 Collaborate to develop a program. [P6] | 1.2.4A A collaboratively created computational artifact reflects effort by more than one person.  1.2.4B Effective collaborative teams consider the use of online collaborative tools.  1.2.4C Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation.  1.2.4D Effective collaboration strategies enhance performance.  1.2.4E Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts.  1.2.4F A collaboratively created computational artifact can reflect personal expressions of ideas.  5.1.3A Collaboration can decrease the size and complexity of tasks required of individual programmers.  5.1.3B Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming.  5.1.3C Collaboration in the iterative development of a program requires different skills than developing a program alone.  5.1.3D Collaboration can make it easier to find and correct errors when developing programs.  5.1.3E Collaboration facilitates developing program components independently.  5.1.3F Effective communication between participants is required for successful collaboration when developing programs. |
| Demonstrate effective interpersonal skills during collaboration. | Effective collaborative teams practice interpersonal skills including but not limited to:   * communication skills; * consensus building; * conflict resolution; and * negotiation. |  | 1.2.4 Collaborate in the creation of computational artifacts. [P6]  3.1.2 Collaborate when processing information to gain insight and knowledge. [P6]  5.1.3 Collaborate to develop a program. [P6] | 1.2.4A A collaboratively created computational artifact reflects effort by more than one person.  1.2.4B Effective collaborative teams consider the use of online collaborative tools.  1.2.4C Effective collaborative teams practice interpersonal communication, consensus building, conflict resolution, and negotiation.  1.2.4D Effective collaboration strategies enhance performance.  1.2.4E Collaboration facilitates the application of multiple perspectives (including sociocultural perspectives) and diverse talents and skills in developing computational artifacts.  1.2.4F A collaboratively created computational artifact can reflect personal expressions of ideas.  3.1.2A Collaboration is an important part of solving data driven problems.  3.1.2B Collaboration facilitates solving computational problems by applying multiple perspectives, experiences, and skill sets.  3.1.2C Communication between participants working on data driven problems gives rise to enhanced insights and knowledge.  3.1.2D Collaboration in developing hypotheses and questions, and in testing hypotheses and answering questions, about data helps participants gain insight and knowledge.  3.1.2E Collaborating face-to-face and using online collaborative tools can facilitate processing information to gain insight and knowledge.  3.1.2F Investigating large data sets collaboratively can lead to insight and knowledge not obtained when working alone.  5.1.3A Collaboration can decrease the size and complexity of tasks required of individual programmers.  5.1.3B Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming.  5.1.3C Collaboration in the iterative development of a program requires different skills than developing a program alone.  5.1.3D Collaboration can make it easier to find and correct errors when developing programs.  5.1.3E Collaboration facilitates developing program components independently.  5.1.3F Effective communication between participants is required for successful collaboration when developing programs. |
| Describe the purpose of a computing innovation. | ~~Computing innovations are developed with the purpose of~~ The purpose of computing innovations is to solve~~ing~~ problems or pursue~~ing~~ interests through creative expression.  An understanding of the purpose of a computing innovation provides developers with an improved ability to develop the computing innovation. | Modified to be less software development process focused based on provider feedback. | 5.1.1 Develop a program for creative expression, to satisfy personal curiosity, or to create new knowledge. [P2] | 5.1.1A Programs are developed and used in a variety of ways by a wide range of people depending on the goals of the programmer.  5.1.1B Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may have visual, audible, or tactile inputs and outputs.  5.1.1C Programs developed for creative expression, to satisfy personal curiosity, or to create new knowledge may be developed with different standards or methods than programs developed for widespread distribution.  5.1.1D Additional desired outcomes may be realized independently of the original purpose of the program.  5.1.1E A computer program or the results of running a program may be rapidly shared with a large number of users and can have widespread impact on individuals, organizations, and society  5.1.1F Advances in computing have generated and increased creativity in other fields. |
| Explain how a program or code segment functions. | * A program is a collection of program statements that performs a specific task when run by a computer. A program is often referred to as software. * A code segment refers to a collection of program statements that are part of a program. * A program needs to work for a variety of inputs and situations. * The behavior of a program is how a program functions during execution and is often described by how a user interacts with it. * A program can be described broadly by what it does or in more detail by both what the program does and how the program statements accomplish this function. |  | **NEW — no 2018 comparison** | |
| Identify input(s) to a program. | Program input is data that are sent to a computer for processing by a program. Input can come in a variety of forms, such as tactile, audible, visual, or text.  An event is associated with an action and supplies input data to a program.  Events can be generated when a key is pressed, a mouse is clicked, a program is started, or by any other defined action that affects the flow of execution.  Inputs usually affect the output produced by a program.    In event-driven programming, program statements are executed when triggered instead of through the sequential flow of control.  Input can come from a user or other applications. |  | 5.1.2 Develop a correct program to solve problems. [P2]  5.2.1 Explain how programs implement algorithms. [P3] | 5.1.2A An iterative process of program development helps in developing a correct program to solve problems.  5.1.2B Developing correct program components and then combining them helps in creating correct programs.  5.1.2C Incrementally adding tested program segments to correct, working programs helps create large correct programs.  5.1.2D Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.  5.1.2E Documentation about program components, such as segments and procedures, helps in developing and maintaining programs.  5.1.2F Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments  5.1.2G Program development includes identifying programmer and user concerns that affect the solution to problems.  5.1.2H Consultation and communication with program users is an important aspect of program development to solve problems.  5.1.2I A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.  5.1.2J A programmer designs, implements, tests, debugs, and maintains programs when solving problems.  5.2.1A Algorithms are implemented using program instructions that are processed during program execution.  5.2.1B Program instructions are executed sequentially.  5.2.1C Program instructions may involve variables that are initialized and updated, read, and written  5.2.1D An understanding of instruction processing and program execution is useful for programming.  5.2.1E Program execution automates processes.  5.2.1F Processes use memory, a central processing unit (CPU), and input and output.  5.2.1G A process may execute by itself or with other processes.  5.2.1H A process may execute on one or several CPUs.  5.2.1I Executable programs increase the scale of problems that can be addressed.  5.2.1J Simple algorithms can solve a large set of problems when automated.  5.2.1K Improvements in algorithms, hardware, and software increase the kinds of problems and the size of problems solvable by programming. |
| Identify output(s) produced by a program. | Program output is any data that are sent from a program to a device. Program output can come in a variety of forms, such as tactile, audible, visual, or text.  Program output is usually based on a program’s input or prior state (such as internal values). |  | 5.1.2 Develop a correct program to solve problems. [P2]  5.2.1 Explain how programs implement algorithms. [P3] | 5.1.2A An iterative process of program development helps in developing a correct program to solve problems.  5.1.2B Developing correct program components and then combining them helps in creating correct programs.  5.1.2C Incrementally adding tested program segments to correct, working programs helps create large correct programs.  5.1.2D Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.  5.1.2E Documentation about program components, such as segments and procedures, helps in developing and maintaining programs.  5.1.2F Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments  5.1.2G Program development includes identifying programmer and user concerns that affect the solution to problems.  5.1.2H Consultation and communication with program users is an important aspect of program development to solve problems.  5.1.2I A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.  5.1.2J A programmer designs, implements, tests, debugs, and maintains programs when solving problems.  5.2.1A Algorithms are implemented using program instructions that are processed during program execution.  5.2.1B Program instructions are executed sequentially.  5.2.1C Program instructions may involve variables that are initialized and updated, read, and written  5.2.1D An understanding of instruction processing and program execution is useful for programming.  5.2.1E Program execution automates processes.  5.2.1F Processes use memory, a central processing unit (CPU), and input and output.  5.2.1G A process may execute by itself or with other processes.  5.2.1H A process may execute on one or several CPUs.  5.2.1I Executable programs increase the scale of problems that can be addressed.  5.2.1J Simple algorithms can solve a large set of problems when automated.  5.2.1K Improvements in algorithms, hardware, and software increase the kinds of problems and the size of problems solvable by programming. |
| Develop a program using a ~~software~~ development process. | A development process could be ordered and intentional, or exploratory in nature.  There are multiple ~~software~~ development processes~~. One that is often used when developing a program includes t~~The following phases are commonly used when developing a program:   * ~~requirements-gathering and analysis~~investigating and reflecting; * designing; * implementating; and * testing.   ~~This course focuses on using a software development process that is incremental and iterative.~~  A ~~software~~ development process that is iterative requires refinement and revision based on feedback, testing or reflection throughout the process. This may require revisiting earlier phases of the process.  A ~~software~~ development process that is incremental is one that breaks the problem into smaller pieces, and makes sure each piece works before adding it to the whole. | Per provider request – we have softened this section so that it isn’t a rigid software development process which would be more than what is expected or intended for an intro course.  changed to investigating so that it is a little lighter. | 1.1.1 Apply a creative development process when creating computational artifacts. [P2]  5.1.2 Develop a correct program to solve problems. [P2] | 1.1.1A A creative process in the development of a computational artifact can include, but is not limited to, employing nontraditional, nonprescribed techniques; the use of novel combinations of artifacts, tools, and techniques; and the exploration of personal curiosities.  1.1.1B Creating computational artifacts employs an iterative and often exploratory process to translate ideas into tangible form.  5.1.2A An iterative process of program development helps in developing a correct program to solve problems.  5.1.2B Developing correct program components and then combining them helps in creating correct programs.  5.1.2C Incrementally adding tested program segments to correct, working programs helps create large correct programs.  5.1.2D Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.  5.1.2E Documentation about program components, such as segments and procedures, helps in developing and maintaining programs.  5.1.2F Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments  5.1.2G Program development includes identifying programmer and user concerns that affect the solution to problems.  5.1.2H Consultation and communication with program users is an important aspect of program development to solve problems.  5.1.2I A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.  5.1.2J A programmer designs, implements, tests, debugs, and maintains programs when solving problems. |
| Design a program and its user interface. | The design of a program incorporates investigation to determine the requirements.  Investigation in a development process is useful in understanding, identifying the constraints, concerns, and interests of the people who will use the program.  Some ways investigation can be performed are as follows:   * + collecting data through surveys,   + user testing,   + interviews, and   + direct observations.   Program requirements describe the function of a program, and may include a description of user interactions that a program must provide.  A program specification defines the requirements for the program.  In a ~~software~~ development process, the design phase outlines how to accomplish a given program specification.  The design phase of a program includes:   * ~~the creation of diagrams that represent the decomposition or modularization of the program;~~ * ~~the creation of diagrams that represent the layouts of the user interface; and~~ * ~~the development of a testing strategy for the program.~~ * brainstorming * planning and storyboarding * organizing the program into modules and functional components * the creation of diagrams that represent the layouts of the user interface; and * the development of a testing strategy for the program. | Additions here meant the deletion of the next 2 LOs. | 5.1.2 Develop a correct program to solve problems. [P2] | 5.1.2A An iterative process of program development helps in developing a correct program to solve problems.  5.1.2B Developing correct program components and then combining them helps in creating correct programs.  5.1.2C Incrementally adding tested program segments to correct, working programs helps create large correct programs.  5.1.2D Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.  5.1.2E Documentation about program components, such as segments and procedures, helps in developing and maintaining programs.  5.1.2F Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments  5.1.2G Program development includes identifying programmer and user concerns that affect the solution to problems.  5.1.2H Consultation and communication with program users is an important aspect of program development to solve problems.  5.1.2I A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.  5.1.2J A programmer designs, implements, tests, debugs, and maintains programs when solving problems. |
| ~~Explain how requirements affect the development of a program.~~ | * ~~The requirements-gathering and analysis phase in a software development process is used to understand and define the purpose of a program, including the data being input and output.~~ * ~~The requirements-gathering and analysis phase can be used to understand the purpose of a program from diverse perspectives in order to develop a program that can be used to fully incorporate these perspectives.~~ * ~~Requirements-gathering and analysis includes identifying any developer or user concerns that affect the program.~~ | Removed to lighten this section per provider feedback | 5.1.2 Develop a correct program to solve problems. [P2] | 5.1.2A An iterative process of program development helps in developing a correct program to solve problems.  5.1.2B Developing correct program components and then combining them helps in creating correct programs.  5.1.2C Incrementally adding tested program segments to correct, working programs helps create large correct programs.  5.1.2D Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.  5.1.2E Documentation about program components, such as segments and procedures, helps in developing and maintaining programs.  5.1.2F Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments  5.1.2G Program development includes identifying programmer and user concerns that affect the solution to problems.  5.1.2H Consultation and communication with program users is an important aspect of program development to solve problems.  5.1.2I A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.  5.1.2J A programmer designs, implements, tests, debugs, and maintains programs when solving problems. |
| ~~Explain how programs meet requirements outlined in program specifications.~~ | * ~~Whether a program meets requirements depends on the context in which it is being used. When considering this context, the usability, functionality and suitability (or appropriateness) of the program is evaluated.~~ * ~~The usability of a program is the ease of understanding and use of the program.~~ * ~~The functionality of a program is the outputs and behaviors based on inputs and user interactions.~~ * ~~The suitability of a program refers to whether it is appropriate for a particular group, purpose, or situation. A program can be used or perceived in an unintended way which can impact whether it is perceived as suitable and meets requirements.~~ * ~~The correctness of a program refers to whether the program will function as expected for all situations.~~ * ~~Correctness can be proven by reasoning formally or mathematically about it or by exhaustively testing every possible case.~~ * ~~Exclusion: Formally proving program correctness is beyond the scope of this course and the AP Exam.~~ * ~~Testing a program is a way to gain confidence in its correctness without proving correctness.~~ | Removed to lighten this section per provider feedback | 5.1.2 Develop a correct program to solve problems. [P2]  5.4.1 Evaluate the correctness of a program. [P4] | 5.1.2A An iterative process of program development helps in developing a correct program to solve problems.  5.1.2B Developing correct program components and then combining them helps in creating correct programs.  5.1.2C Incrementally adding tested program segments to correct, working programs helps create large correct programs.  5.1.2D Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.  5.1.2E Documentation about program components, such as segments and procedures, helps in developing and maintaining programs.  5.1.2F Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments  5.1.2G Program development includes identifying programmer and user concerns that affect the solution to problems.  5.1.2H Consultation and communication with program users is an important aspect of program development to solve problems.  5.1.2I A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.  5.1.2J A programmer designs, implements, tests, debugs, and maintains programs when solving problems.  5.4.1A Program style can affect the determination of program correctness.  5.4.1B Duplicated code can make it harder to reason about a program.  5.4.1C Meaningful names for variables and procedures help people better understand programs.  5.4.1D Longer code segments are harder to reason about than shorter code segments in a program.  5.4.1E Locating and correcting errors in a program is called debugging the program.  5.4.1F Knowledge of what a program is supposed to do is required in order to find most program errors.  5.4.1G Examples of intended behavior on specific inputs help people understand what a program is supposed to do.  5.4.1H Visual displays (or different modalities) of program state can help in finding errors.  5.4.1I Programmers justify and explain a programâ™s correctness.  5.4.1J Justification can include a written explanation about how a program meets its specifications.  5.4.1K Correctness of a program depends on correctness of program components, including code segments and procedures.  5.4.1L An explanation of a program helps people understand the functionality and purpose of it.  5.4.1M The functionality of a program is often described by how a user interacts with it.  5.4.1N The functionality of a program is best described at a high level by what the program does, not at the lower level of how the program statements work to accomplish this. |
| Describe the purpose of a code segment or program by writing documentation. | * Program documentation is a written description of the function of a code ~~segment’s or program’s function and how it was developed.~~segment, event, procedure, or program and how it was developed. * Comments are a form of documentation written into the program to be read by people and do not impact how a program runs. * Programmers should document a program throughout the process of developing a program. * Program documentation ~~about program components, such as code segments, events, and procedures,~~ helps in developing and maintaining correct programs when working individually or in collaborative programming environments. * Not all programming environments support comments, so other methods of documentation may be required. |  | 5.1.2 Develop a correct program to solve problems. [P2] | 5.1.2A An iterative process of program development helps in developing a correct program to solve problems.  5.1.2B Developing correct program components and then combining them helps in creating correct programs.  5.1.2C Incrementally adding tested program segments to correct, working programs helps create large correct programs.  5.1.2D Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.  5.1.2E Documentation about program components, such as segments and procedures, helps in developing and maintaining programs.  5.1.2F Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments  5.1.2G Program development includes identifying programmer and user concerns that affect the solution to problems.  5.1.2H Consultation and communication with program users is an important aspect of program development to solve problems.  5.1.2I A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.  5.1.2J A programmer designs, implements, tests, debugs, and maintains programs when solving problems. |
| Acknowledge code segments used from other sources. | It is important to acknowledge any code segments that were developed collaboratively or by another source.  Acknowledgement of code segment(s) written by someone else and used in a program can be in the program documentation. The acknowledgement should include the origin or original author’s name. |  | 5.1.2 Develop a correct program to solve problems. [P2]  7.5.1 Access, manage, and attribute information using effective strategies. [P1] | 5.1.2A An iterative process of program development helps in developing a correct program to solve problems.  5.1.2B Developing correct program components and then combining them helps in creating correct programs.  5.1.2C Incrementally adding tested program segments to correct, working programs helps create large correct programs.  5.1.2D Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.  5.1.2E Documentation about program components, such as segments and procedures, helps in developing and maintaining programs.  5.1.2F Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments  5.1.2G Program development includes identifying programmer and user concerns that affect the solution to problems.  5.1.2H Consultation and communication with program users is an important aspect of program development to solve problems.  5.1.2I A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.  5.1.2J A programmer designs, implements, tests, debugs, and maintains programs when solving problems.  7.5.1A Online databases and libraries catalog and house secondary and some primary resources  7.5.1B Advanced search tools, Boolean logic, and key words can refine the search focus and/or limit search results based on a variety of factors (e.g., peer-review status, type of publication  7.5.1C Plagiarism is a serious offense that occurs when a person presents another's ideas or words as his or her own. Plagiarism may be avoided by accurately acknowledging sources. |
| For errors in a program:  (a) Identify the error.  (b) Correct the error. | A logic error is a mistake in the program that allows a program to run but causes it to behave incorrectly or unexpectedly.  A syntax error is a mistake in the program where the rules of the programming language are not followed.  A run-time error is a mistake in the program that occurs during the execution of a program. Programming languages define their own run-time errors.  An overflow error is an error that occurs when the computer attempts to handle a number that is outside of the defined range of values can be represented.  The following are effective ways to find and correct errors:   * test-cases; * hand-tracing; * visualizations; * debuggers; and * adding extra output statement. |  | 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. [P4]  5.1.2 Develop a correct program to solve problems. [P2]  5.4.1 Evaluate the correctness of a program. [P4] | 1.2.5A The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact.  1.2.5B A computational artifact may have weaknesses, mistakes, or errors depending on the type of artifact.  1.2.5C The functionality of a computational artifact may be related to how it is used or perceived.  1.2.5D The suitability (or appropriateness) of a computational artifact may be related to how it is used or perceived.  5.1.2A An iterative process of program development helps in developing a correct program to solve problems.  5.1.2B Developing correct program components and then combining them helps in creating correct programs.  5.1.2C Incrementally adding tested program segments to correct, working programs helps create large correct programs.  5.1.2D Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.  5.1.2E Documentation about program components, such as segments and procedures, helps in developing and maintaining programs.  5.1.2F Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments  5.1.2G Program development includes identifying programmer and user concerns that affect the solution to problems.  5.1.2H Consultation and communication with program users is an important aspect of program development to solve problems.  5.1.2I A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.  5.1.2J A programmer designs, implements, tests, debugs, and maintains programs when solving problems.  5.4.1A Program style can affect the determination of program correctness.  5.4.1B Duplicated code can make it harder to reason about a program.  5.4.1C Meaningful names for variables and procedures help people better understand programs.  5.4.1D Longer code segments are harder to reason about than shorter code segments in a program.  5.4.1E Locating and correcting errors in a program is called debugging the program.  5.4.1F Knowledge of what a program is supposed to do is required in order to find most program errors.  5.4.1G Examples of intended behavior on specific inputs help people understand what a program is supposed to do.  5.4.1H Visual displays (or different modalities) of program state can help in finding errors.  5.4.1I Programmers justify and explain a programâ™s correctness.  5.4.1J Justification can include a written explanation about how a program meets its specifications.  5.4.1K Correctness of a program depends on correctness of program components, including code segments and procedures.  5.4.1L An explanation of a program helps people understand the functionality and purpose of it.  5.4.1M The functionality of a program is often described by how a user interacts with it.  5.4.1N The functionality of a program is best described at a high level by what the program does, not at the lower level of how the program statements work to accomplish this. |
| Identify ~~test cases and expected outcomes for a program.~~inputs and corresponding expected output or behaviors that can be used to check the correctness of a program. | In the ~~software~~ development process, testing uses defined ~~test cases~~ inputs to ensure that a program is producing the expected outcomes. Programmers use the result from testing to revise their programs.  ~~The test cases for a program consist of potential inputs and the resulting outputs.~~  ~~Test cases should include inputs that~~ Defined inputs used to test a program should demonstrate the different expected outcomes ~~and~~ that are at or just beyond the extremes, minimum and maximum, of input data.    Program requirements are needed to identify ~~test cases~~ appropriate defined inputs for testing. | The use of the term test cases, lead people to think of unit test cases. We are now describing what we meant by “test case” in terms of an input and expected output. | 1.2.5 Analyze the correctness, usability, functionality, and suitability of computational artifacts. [P4]  5.1.2 Develop a correct program to solve problems. [P2] | 1.2.5A The context in which an artifact is used determines the correctness, usability, functionality, and suitability of the artifact.  1.2.5B A computational artifact may have weaknesses, mistakes, or errors depending on the type of artifact.  1.2.5C The functionality of a computational artifact may be related to how it is used or perceived.  1.2.5D The suitability (or appropriateness) of a computational artifact may be related to how it is used or perceived.  5.1.2A An iterative process of program development helps in developing a correct program to solve problems.  5.1.2B Developing correct program components and then combining them helps in creating correct programs.  5.1.2C Incrementally adding tested program segments to correct, working programs helps create large correct programs.  5.1.2D Program documentation helps programmers develop and maintain correct programs to efficiently solve problems.  5.1.2E Documentation about program components, such as segments and procedures, helps in developing and maintaining programs.  5.1.2F Documentation helps in developing and maintaining programs when working individually or in collaborative programming environments  5.1.2G Program development includes identifying programmer and user concerns that affect the solution to problems.  5.1.2H Consultation and communication with program users is an important aspect of program development to solve problems.  5.1.2I A programmer's knowledge and skill affects how a program is developed and how it is used to solve a problem.  5.1.2J A programmer designs, implements, tests, debugs, and maintains programs when solving problems. |

Big Idea 2: Data

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2020 | | Rationale for changes | 2018 | | |
| Learning Objectives | Essential Knowledge |  | Learning Objectives | Essential Knowledge |
| Students must be able to… | Students will know that… |  | Students must be able to… | Students will know that… |
| Explain how data can be represented using bits. | * Data ~~is~~are ~~the~~ values that can be stored in ~~of~~ variables, passed as input to procedures and returned from procedures, items of lists, or standalone constant values. * ~~Information is the collection of facts and patterns extracted from data.~~ * ~~Data on a~~ Computing devices ~~is~~ represent~~ed~~ data digitally, which means ~~using digital data. All digital data are represented at~~ the lowest level in the computer ~~using~~components of any values are bits. * A “bit” is shorthand for “binary digit”, and is either 0 or 1. * A “byte” is 8 bits. ~~A byte is a group of digital data containing 8 bits.~~ * Abstraction is the process of reducing complexity by focusing on the main idea. By hiding details irrelevant to the question at hand and bringing together related and useful detail, abstraction reduces complexity and allows one to focus on the ~~problem~~idea. * ~~At a higher level,~~ Bits are grouped to represent abstractions.~~,~~ These abstraction include,~~ing~~ but not limited to numbers, characters, and color. * The same sequence of bits may represent different types of data in different contexts. * ~~Analog data is real-world data, such as your voice, and is stored in a physical way, such as grooves etched into vinyl or magnetic tapes.~~ * ~~Analog data can be stored as digital data in a process which approximates real-world data using a sampling technique.~~ * ~~Sampling involves measuring values of the analog signal at regular intervals (usually in time or space) called samples.~~ * ~~The samples are measured to figure out the exact bits required to store each sample.~~ * Analog data are data with values that change continuously, or smoothly, over time. Some examples of analog data include music, colors of a painting, or position of a sprinter during a race. * The use of digital data to approximate real-world analog data is an example of abstraction. * Analog data can be closely approximated digitally using a sampling technique, which means measuring values of the analog signal at regular intervals (usually in time or space) called samples. The samples are measured to figure out the exact bits required to store each sample. | The committee felt this LO needed to be split, as the Eks seemed to be about 2 different things we would want students to know how to do. | 2.1.1 Describe the variety of abstractions used to represent data. [P3]  2.1.2 Explain how binary sequences are used to represent digital data. [P5] | 2.1.1A Digital data is represented by abstractions at different levels.  2.1.1B At the lowest level, all digital data are represented by bits.  2.1.1C At a higher level, bits are grouped to represent abstractions, including but not limited to numbers, characters, and color.  2.1.1D Number bases, including binary, decimal, and hexadecimal, are used to represent and investigate digital data.  2.1.1E At one of the lowest levels of abstraction, digital data is represented in binary (base 2) using only combinations of the digits zero and one.  Exclusion Statement (2.1.1E): Two's complement conversions are beyond the scope of this course and the AP Exam.  2.1.1F Hexadecimal (base 16) is used to represent digital data because hexadecimal representation uses fewer digits than binary.  2.1.1G Numbers can be converted from any base to any other base.  2.1.2A A finite representation is used to model the infinite mathematical concept of a number.  2.1.2B In many programming languages, the fixed number of bits used to represent characters or integers limits the range of integer values and mathematical operations; this limitation can result in overflow or other errors.  Exclusion Statement (2.1.2B): Range limitations of any one language, compiler, or architecture are beyond the scope of this course and the AP Exam.  2.1.2C In many programming languages, the fixed number of bits used to represent real numbers (as floating point numbers) limits the range of floating point values and mathematical operations; this limitation can result in round  2.1.2D The interpretation of a binary sequence depends on how it is used.  2.1.2E A sequence of bits may represent instructions or data.  2.1.2F A sequence of bits may represent different types of data in different contexts. |
| Explain the consequences of using bits to represent data. | * In many programming languages, the fixed number of bits used to represent integers limits the range of integer values and mathematical operations on those values; this limitation can result in overflow or other errors. * Other programming languages provide an abstraction through which the size of representable integers is limited only by the size of the computer’s memory; this is the case for the language defined in the exam reference sheet * In many programming languages the fixed number of bits used to represent real numbers limits the range of values and mathematical operations on those values; this limitation can result in round-off and other errors. ~~Some r~~Real numbers can only be approximated in computer storage.EXCLUSION STATEMENT (for EK): Specific range limitations are beyond the scope of this course and the AP Exam. |
| For binary numbers:  (a) Calculate the binary (base 2) equivalent of a positive integer (base 10) and vice versa.  (b) Compare and order binary numbers. | * Number bases, including binary and decimal, are used to represent data. * Binary (base 2) uses only combinations of the digits zero and one. * Decimal (base 10) uses only combinations of the digits 0 – 9. * As with decimal, a digit’s position within the binary sequence determines its numeric value. The numeric value is equal to the bits value multiplied by the place value of the position. * The place value of each position is determined by the base raised to the power of the position. Positions are numbered starting at the right most position with 0 and increasing by 1 for each subsequent position to the left. |  | 2.1.1 Describe the variety of abstractions used to represent data. [P3]  2.1.2 Explain how binary sequences are used to represent digital data. [P5] | 2.1.1A Digital data is represented by abstractions at different levels.  2.1.1B At the lowest level, all digital data are represented by bits.  2.1.1C At a higher level, bits are grouped to represent abstractions, including but not limited to numbers, characters, and color.  2.1.1D Number bases, including binary, decimal, and hexadecimal, are used to represent and investigate digital data.  2.1.1E At one of the lowest levels of abstraction, digital data is represented in binary (base 2) using only combinations of the digits zero and one.  Exclusion Statement (2.1.1E): Two's complement conversions are beyond the scope of this course and the AP Exam.  2.1.1F Hexadecimal (base 16) is used to represent digital data because hexadecimal representation uses fewer digits than binary.  2.1.1G Numbers can be converted from any base to any other base.  2.1.2A A finite representation is used to model the infinite mathematical concept of a number.  2.1.2B In many programming languages, the fixed number of bits used to represent characters or integers limits the range of integer values and mathematical operations; this limitation can result in overflow or other errors.  Exclusion Statement (2.1.2B): Range limitations of any one language, compiler, or architecture are beyond the scope of this course and the AP Exam.  2.1.2C In many programming languages, the fixed number of bits used to represent real numbers (as floating point numbers) limits the range of floating point values and mathematical operations; this limitation can result in round  2.1.2D The interpretation of a binary sequence depends on how it is used.  2.1.2E A sequence of bits may represent instructions or data.  2.1.2F A sequence of bits may represent different types of data in different contexts. |
| Compare data compression algorithms to determine which is best in a particular context. | * Data compression can reduce the amount of data (number of bits) transmitted or stored. * Fewer bits does not necessarily mean less information. * The amount of reduction in bits by compression depends on both the ~~arrangement of bits in the original data~~amount of redundancy in the original representation and the compression algorithm applied. * Lossless data compression algorithms can usually reduce the number of bits stored or transmitted~~,~~ while guaranteeing complete reconstruction of the original data. * Lossy data compression algorithms can significantly reduce the number of bits stored or transmitted, but only allow the reconstruction of an approximation of the original data. * Lossy data compression algorithms usually reduce the number of bits stored or transmitted more than lossless compression. * In situations where quality or ability to reconstruct the original is maximally important, lossless compression algorithms are typically chosen. * In situations where minimizing data size or short transmission time is maximally important, lossy compression algorithms are typically chosen. |  | 3.3.1 Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information. [P4] | 3.3.1C There are trade offs in using lossy and lossless compression techniques for storing and transmitting data.  3.3.1D Lossless data compression reduces the number of bits stored or transmitted but allows complete reconstruction of the original data  3.3.1E Lossy data compression can significantly reduce the number of bits stored or transmitted at the cost of being able to reconstruct only an approximation of the original data. |
| Describe what information can be extracted from data. | * Information is the collection of facts and patterns extracted from data. * Data provide opportunities for identifying trends, making connections, and addressing problems. * A correlation is an association between two or more things. * Digitally processed data may show correlation between variables. A correlation found in data does not necessarily imply a causation relationship exists. Often additional research is needed to verify causation. * Often a single data source does not contain the necessary data to draw a conclusion. It may be required to combine data from a variety sources to formulate a conclusion. |  | 3.2.1 Extract information from data to discover and explain connections or trends. [P1] | 3.2.1A Large data sets provide opportunities and challenges for extracting information and knowledge.  3.2.1B Large data sets provide opportunities for identifying trends, making connections in data, and solving problems.  3.2.1C Computing tools facilitate the discovery of connections in information within large data sets.  3.2.1D Search tools are essential for efficiently finding information.  3.2.1E Information filtering systems are important tools for finding information and recognizing patterns in the information.  3.2.1F Software tools, including spreadsheets and databases, help to efficiently organize and find trends in information.  Exclusion Statement (3.2.1F): Students are not expected to know specific formulas or options available in spreadsheet or database software packages..  3.2.1G Metadata is data about data.  3.2.1H Metadata can be descriptive data about an image, a Web page, or other complex objects.  3.2.1I Metadata can increase the effective use of data or data sets by providing additional information about various aspects of that data. |
| Describe what information can be extracted from metadata. | * Metadata is data about data. ~~It can be~~ Metadata is associated with the primary data; the primary data may be~~, such as~~ an image, a Web page, or another complex object. * Changes and deletions made to metadata do not change the primary data. * Metadata ~~is~~are used for finding, organizing and managing information. * Metadata can increase the effective use of data or data sets by providing additional information ~~about various aspects of that data~~. * Metadata allows data to be structured and organized. |  | 3.2.1 Extract information from data to discover and explain connections or trends. [P1] | 3.2.1A Large data sets provide opportunities and challenges for extracting information and knowledge.  3.2.1B Large data sets provide opportunities for identifying trends, making connections in data, and solving problems.  3.2.1C Computing tools facilitate the discovery of connections in information within large data sets.  3.2.1D Search tools are essential for efficiently finding information.  3.2.1E Information filtering systems are important tools for finding information and recognizing patterns in the information.  3.2.1F Software tools, including spreadsheets and databases, help to efficiently organize and find trends in information.  Exclusion Statement (3.2.1F): Students are not expected to know specific formulas or options available in spreadsheet or database software packages..  3.2.1G Metadata is data about data.  3.2.1H Metadata can be descriptive data about an image, a Web page, or other complex objects.  3.2.1I Metadata can increase the effective use of data or data sets by providing additional information about various aspects of that data. |
| Identify the challenges associated with processing data. | * The ability to process data depends on the capabilities of the users and their tools. Therefore, a data set can be considered to be a large data set based on the capabilities of some users, but not necessarily for all users. * Data pose challenges regardless of size. Such as:   + the need to clean data   + incomplete data   + invalid data   + the need to combine data sources * Depending on how the data is being collected, the data may not be uniform. For example, if users enter data into an open field, the way they choose to abbreviate, spell or capitalize something may vary from user to user. * Cleaning data is a process that makes the data uniform~~,~~ without changing its meaning. For example, replacing all abbreviations, spellings and capitalizations with the same word. * Problems of bias are often caused by the type or source of data that is being collected. Bias is not eliminated by simply collecting more data. * The size of the data set affects the amount of information that can be extracted from it. * Large data sets are difficult to process using a single computer and may require parallel systems. * Scalability of systems is an important consideration when working with large data sets, since the size or scale of a system affects how that data set can be processed and stored. |  | 3.1.1 Find patterns, and test hypotheses about digitally processed information to gain insight and knowledge. [P4]  3.2.2 Determine how large data sets impact the use of computational processes to discover information and knowledge. [P3] | 3.1.1A Computers are used in an iterative and interactive way when processing digital information to gain insight and knowledge.  3.1.1B Digital information can be filtered and cleaned by using computers to process information.  3.1.1C Combining data sources, clustering data, and data classification are part of the process of using computers to process information.  3.1.1D Insight and knowledge can be obtained from translating and transforming digitally represented information.  3.1.1E Patterns can emerge when data is transformed using computational tools.  3.2.2A Large data sets include data such as transactions, measurements, text, sound, images, and video.  3.2.2B The storing, processing, and curating of large data sets is challenging.  3.2.2C Structuring large data sets for analysis can be challenging.  3.2.2D Maintaining privacy of large data sets containing personal information can be challenging.  3.2.2E Scalability of systems is an important consideration when data sets are large.  3.2.2F The size or scale of a system that stores data affects how that data set is used.  3.2.2G The effective use of large data sets requires computational solutions.  3.2.2H Analytical techniques to store, manage, transmit, and process data sets change as the size of data sets scale. |
| Extract information from data using a program. | Programs can be used to process data to acquire information.  Tables, diagrams, and textual displays can be used in communicating insight and knowledge gained from data.  Search tools are useful for efficiently finding information.  Data filtering systems are important tools for finding information and recognizing patterns in the data.  Programs, **including spreadsheets**, help to efficiently organize and find trends in information.  Some processes that can be used to extract or modify information from data include:   * transforming every element of a data set, such as doubling every element in a list, or extracting the parent’s email from every student record; * filtering a data set, such as keeping only positive numbers from a list of numbers, or keeping only students who signed up for band from all the student records; * combining or comparing data in some way, such as adding up a list of numbers, or finding the student who has the highest GPA; OR * visualize data set through a chart, graph or other visual representation. |  | 3.1.1 Find patterns, and test hypotheses about digitally processed information to gain insight and knowledge. [P4]  3.1.3 Explain the insight and knowledge gained from digitally processed data by using appropriate visualizations, notations, and precise language. [P5]  3.2.1 Extract information from data to discover and explain connections or trends. [P1] | 3.1.1A Computers are used in an iterative and interactive way when processing digital information to gain insight and knowledge.  3.1.1B Digital information can be filtered and cleaned by using computers to process information.  3.1.1C Combining data sources, clustering data, and data classification are part of the process of using computers to process information.  3.1.1D Insight and knowledge can be obtained from translating and transforming digitally represented information.  3.1.1E Patterns can emerge when data is transformed using computational tools.  3.1.3A Visualization tools and software can communicate information about data.  3.1.3B Tables, diagrams, and textual displays can be used in communicating insight and knowledge gained from data.  3.1.3C Summaries of data analyzed computationally can be effective in communicating insight and knowledge gained from digitally represented information.  3.1.3D Transforming information can be effective in communicating knowledge gained from data.  3.1.3E Interactivity with data is an aspect of communicating.  3.2.1A Large data sets provide opportunities and challenges for extracting information and knowledge.  3.2.1B Large data sets provide opportunities for identifying trends, making connections in data, and solving problems.  3.2.1C Computing tools facilitate the discovery of connections in information within large data sets.  3.2.1D Search tools are essential for efficiently finding information.  3.2.1E Information filtering systems are important tools for finding information and recognizing patterns in the information.  3.2.1F Software tools, including spreadsheets and databases, help to efficiently organize and find trends in information.  Exclusion Statement (3.2.1F): Students are not expected to know specific formulas or options available in spreadsheet or database software packages..  3.2.1G Metadata is data about data.  3.2.1H Metadata can be descriptive data about an image, a Web page, or other complex objects.  3.2.1I Metadata can increase the effective use of data or data sets by providing additional information about various aspects of that data. |
| Explain how programs can be used to gain insight and knowledge from data. | * Programs are used in an iterative and interactive way when processing information to allow users to gain insight and knowledge. * Programmers can use programs to filter and clean digital data ~~by using programs to~~ , thereby gaining insight and knowledge. * Combining data sources, clustering data, and classifying data are parts of the process of using programs to gain insight and knowledge from data. * Insight and knowledge can be obtained from translating and transforming digitally represented information. * Patterns can emerge when data is transformed using programs. | Added: use programs (excel is a program, so that would be allowed) | 3.1.1 Find patterns, and test hypotheses about digitally processed information to gain insight and knowledge. [P4] | 3.1.1A Computers are used in an iterative and interactive way when processing digital information to gain insight and knowledge.  3.1.1B Digital information can be filtered and cleaned by using computers to process information.  3.1.1C Combining data sources, clustering data, and data classification are part of the process of using computers to process information.  3.1.1D Insight and knowledge can be obtained from translating and transforming digitally represented information.  3.1.1E Patterns can emerge when data is transformed using computational tools. |

Big Idea 3: Algorithms and Programming

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2020 | | Rationale for changes | 2018 | |
| Learning Objectives | Essential Knowledge |  | Learning Objectives | Essential Knowledge |
| Students must be able to… | Students will know that… |  | Students must be able to… | Students will know that… |
|  | “Distributed throughout” |  | 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. [P2]  2.2.2 Use multiple levels of abstraction to write programs. [P3] | 2.2.1A The process of developing an abstraction involves removing detail and generalizing functionality.  2.2.1B An abstraction extracts common features from specific examples in order to generalize concepts.  2.2.1C An abstraction generalizes functionality with input parameters that allow software reuse.  Exclusion Statement: (2.2.1C): An understanding of the difference between value and reference parameters is beyond the scope of this course and the AP Exam.  2.2.2A Software is developed using multiple levels of abstractions, such as constants, expressions, statements, procedures, and libraries.  2.2.2B Being aware of and using multiple levels of abstraction in developing programs helps to more effectively apply available resources and tools to solve problems. |
|  | PROPOSED TO REMOVE logic gates, chips and hardware components |  | 2.2.3 Identify multiple levels of abstractions that are used when writing programs. [P3] | 2.2.3A Different programming languages offer different levels of abstraction.  Exclusion Statement (2.2.3A): Knowledge of the abstraction capabilities of all programming languages is beyond the scope of this course and the AP Exam.  2.2.3B High level programming languages provide more abstractions for the programmer and make it easier for people to read and write a program.  2.2.3C Code in a programming language is often translated into code in another (lowerlevel) language to be executed on a computer.  2.2.3D In an abstraction hierarchy, higher levels of abstraction (the most general concepts) would be placed toward the top and lower level abstractions (the more specific concepts) toward the bottom.  2.2.3E Binary data is processed by physical layers of computing hardware, including gates, chips, and components.  2.2.3F A logic gate is a hardware abstraction that is modeled by a Boolean function.  Exclusion Statement (2.2.3F): Memorization of specific gate visual representations is beyond the scope of this course and the AP Exam.  2.2.3G A chip is an abstraction composed of low level components and circuits that perform a specific function.  2.2.3H A hardware component can be low level like a transistor or high level like a video card.  2.2.3I Hardware is built using multiple levels of abstractions, such as transistors, logic gates, chips, memory, motherboards, special purposes cards, and storage devices.  2.2.3J Applications and systems are designed, developed, and analyzed using levels of hardware, software, and conceptual abstractions.  2.2.3K Lowerlevel abstractions can be combined to make higher level abstractions, such as short message services (SMS) or email messages, images, audio files, and videos. |
|  | REMOVED |  | 4.2.2 Explain the difference between solvable and unsolvable problems in computer science. [P1] | Exclusion Statement (LO 4.2.2): Determining whether a given problem is olvable or unsolvable is beyond the scope of this course and the AP Exam.  4.2.2A A heuristic is a technique that may allow us to find an approximate solution when typical methods fail to find an exact solution.  4.2.2B Heuristics may be helpful for finding an approximate solution more quickly when exact methods are too slow.  Exclusion Statement (4.2.2B): Specific heuristic solutions are beyond the scope of this course and the AP Exam.  4.2.2C Some optimization problems such as 'find the best' or 'find the smallest' cannot be solved in a reasonable time, but approximations to the optimal solution can.  4.2.2D Some problems cannot be solved using any algorithm. |
|  | REMOVED? BROOK AND GT COULDN’T FIND WHEN READING THROUGH NEW EKS  \*\*\* This is in Big Idea 1, EU 1, LO 1 above. I’ve copied it there - CRYSTAL | Please see note in the left column. In addition to there being LOs on collaboration, we will further emphasize the importance of collaboration through the computational thinking practices. | 5.1.3 Collaborate to develop a program. [P6] | 5.1.3A Collaboration can decrease the size and complexity of tasks required of individual programmers.  5.1.3B Collaboration facilitates multiple perspectives in developing ideas for solving problems by programming.  5.1.3C Collaboration in the iterative development of a program requires different skills than developing a program alone.  5.1.3D Collaboration can make it easier to find and correct errors when developing programs.  5.1.3E Collaboration facilitates developing program components independently.  5.1.3F Effective communication between participants is required for successful collaboration when developing programs. |
| Explain the existence of undecidable problems in computer science. | A decidable problem is a decision problem ~~one in~~ for which an algorithm can be ~~constructed to provide a yes-or-no answer for all inputs (e.g., “Is the number even?”).~~ written to produce a correct output for all inputs (e.g., “Is the number even?”).  An undecidable problem is one in which no algorithm can be constructed that is always capable of providing a correct yes-or-no answer.  EXCLUSION STATEMENT (for EK 4.2.3C): Determining whether a given problem is undecidable is beyond the scope of this course and the AP Exam.  An undecidable problem may have some instances that have an algorithmic solution, but there is no algorithmic solution that solves all instances of the problem. | This section is move to the end of this big idea.  Additionally this will be the 2nd LO for this EU. | 4.2.3 Explain the existence of undecidable problems in computer science. [P1] | 4.2.3A An undecidable problem may have instances that have an algorithmic solution, but there is no algorithmic solution that solves all instances of the problem.  4.2.3B A decidable problem is one in which an algorithm can be constructed to answer 'yes' or 'no' for all inputs (e.g., 'is the number even')  4.2.3C An undecidable problem is one in which no algorithm can be constructed that always leads to a correct yes or no answer  Exclusion Statement (4.2.3C): Determining whether a given problem is undecidable is beyond the scope of this course and the AP Exam. |
| For determining ~~whether an algorithm runs in reasonable time~~the efficiency of an algorithm:  (a)Explain the difference between algorithms that run in reasonable time and those that do not run in reasonable time.  (b) Identify situations where a heuristic solution may be more appropriate. | * A problem is a general description of a task that may (or may not) be solved algorithmically. An instance of a problem also includes specific input. For example, sorting is a problem, sorting the list (2,3,1,7) is an instance of the problem. * A decision problem is a problem with a yes-no answer. An optimization problem is a problem with the goal of finding the "best" solution among many. For example, is there a path from A to B? What is the shortest path from A to B? * ~~Efficiency is the ratio of the amount of output produced to the amount of input used.~~ * ~~Different correct algorithms for the same problem can have different efficiencies.~~ * Efficiency ~~includes both execution time and memory usage~~ measures the number of steps an algoirthm requires before it terminates. Efficiency is a function of the size of the input. * EXCLUSION STATEMENT: Formal analysis of algorithms (Big-O) and formal reasoning using mathematical formulas are beyond the scope of this course and the AP Exam. * Determining an algorithm’s efficiency is done by reasoning formally or mathematically about the algorithm. * ~~A way to gauge if one solution would be considered more efficient than another without measuring execution time and memory usage would be to see if the solution can be accomplished in fewer steps.~~ An algorithm’s efficiency can be informally measured by determining the number of times a statement or group of statements executes. * Different correct algorithms for the same problem can have different efficiencies. * ~~For a problem to be solved in reasonable time means that the number of steps the algorithm takes is less than or equal to a polynomial function~~ Algorithms with efficiencies that grow at a polynomial rate or slower (constant, linear, square, cube, etc.) are said to run in a reasonable amount of time. ~~of the size of the input~~.EXCLUSION STATEMENT (for EK ): Using non-polynomial functions to describe relationships between the number of steps required by an algorithm and the input size is beyond the scope of this course and the AP Exam. * Some problems cannot be solved in a reasonable time~~, even for small input sizes.~~ because there is no efficient algorithm for solving them. In these cases, ~~heuristic approaches may be helpful to find an approximate solution in reasonable time.~~approximate solutions are sought. * A heuristic is a technique that may allow programmers to find an approximate solution when typical methods fail to find an exact solution. EXCLUSION STATEMENT (for EK 4.2.2B): Specific heuristic solutions are beyond the scope of this course and the AP Exam. * ~~Some optimization problems such as “find the best” or “find the smallest” cannot be solved in a reasonable time but approximations to the optimal solution can.~~ | This section is move to the end of this big idea.  Additionally this is the 1st LO for this EU. | 4.2.1 Explain the difference between algorithms that run in a reasonable time and those that do not run in a reasonable time. [P1]  4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity. [P4] | Exclusion Statement (LO 4.2.1): Any discussion of nondeterministic polynomial (NP) is beyond the scope of this course and the AP Exam.  4.2.1A Many problems can be solved in a reasonable time.  4.2.1B Reasonable time means that the number of steps the algorithm takes is less than or equal to a polynomial function (constant, linear, square, cube, etc.) of the size of the input.  Exclusion Statement (4.2.1B): Using non-polynomial functions to describe relationships between the number of steps required by an algorithm and the input size is beyond the scope of this course and the AP Exam.  4.2.1C Some problems cannot be solved in a reasonable time, even for small input sizes.  4.2.1D Some problems can be solved but not in a reasonable time. In these cases, heuristic approaches may be helpful to find solutions in reasonable time.  4.2.4A Determining an algorithm's efficiency is done by reasoning formally or mathematically about the algorithm.  4.2.4B Empirical analysis of an algorithm is done by implementing the algorithm and running it on different inputs.  4.2.4C The correctness of an algorithm is determined by reasoning formally or mathematically about the algorithm, not by testing an implementation of the algorithm.  Exclusion Statement (4.2.4C): Formally proving program correctness is beyond the scope of this course and the AP Exam.  4.2.4D Different correct algorithms for the same problem can have different efficiencies.  4.2.4E Sometimes more efficient algorithms are more complex.  4.2.4F Finding an efficient algorithm for a problem can help solve larger instances of the problem.  4.2.4G Efficiency includes both execution time and memory usage.  Exclusion Statement (4.2.4G): Formal analysis of algorithms (BigO) and formal reasoning using mathematical formulas are beyond the scope of this course and the AP Exam.  4.2.4H Linear search can be used when searching for an item in any list; binary search can be used only when the list is sorted. |
| ~~Define a variable to store a value.~~  Represent a value with a variable. | * A variable is an abstraction ~~used to name a data storage location~~ inside the program that can hold a value. Each variable has associated data storage that represents one value at a time, but that value can be a list or other collection that in turn contains multiple vlaues. ~~is used to hold its value.~~ * Using meaningful variable names helps ~~the reader~~ computer scientists understand what values are ~~stored in~~ represented by the variable. * ~~A meaningful variable name is descriptive of what it is storing.~~ * ~~P~~ Some programming languages provide types to represent data, which are referenced using variables. Some types include: numbers, Booleans, lists, and strings. * Some ~~problems are more easily solved with one type of data than another~~values are better suited to representation using one type of data rather than another. * ~~Abstraction allows the use of integers and real numbers in programs without requiring understanding of how they are implemented~~. |  | 2.2.2 Use multiple levels of abstraction to write programs. [P3] | 2.2.2A Software is developed using multiple levels of abstractions, such as constants, expressions, statements, procedures, and libraries.  2.2.2B Being aware of and using multiple levels of abstraction in developing programs helps to more effectively apply available resources and tools to solve problems. |
| Determine the value ~~that is stored in a~~ of a variable as a result of an assignment. | The assignment operator allows a program to change the value ~~stored in a~~ represented by a variable. ~~The value on the right is evaluated and stored in the variable on the left.~~ The exam reference sheet provides ← operator to use for assignment. For example, playerName ← expression which evaluates expression and assigns the result to the variable playerName.  The value stored in a variable will be the most recent value assigned. For example: a<- 1; b <- a; a <- 2; display (b) still displays 1. |  | 5.2.1 Explain how programs implement algorithms. [P3]  5.3.1 Use abstraction to manage complexity in programs. [P3] | 5.2.1A Algorithms are implemented using program instructions that are processed during program execution.  5.2.1B Program instructions are executed sequentially.  5.2.1C Program instructions may involve variables that are initialized and updated, read, and written  5.2.1D An understanding of instruction processing and program execution is useful for programming.  5.2.1E Program execution automates processes.  5.2.1F Processes use memory, a central processing unit (CPU), and input and output.  5.2.1G A process may execute by itself or with other processes.  5.2.1H A process may execute on one or several CPUs.  5.2.1I Executable programs increase the scale of problems that can be addressed.  5.2.1J Simple algorithms can solve a large set of problems when automated.  5.2.1K Improvements in algorithms, hardware, and software increase the kinds of problems and the size of problems solvable by programming.  5.3.1A Procedures are reusable programming abstractions.  5.3.1B A function is a named grouping of programming instructions.  5.3.1C Procedures reduce the complexity of writing and maintaining programs.  5.3.1D Procedures have names and may have parameters and return values.  5.3.1E Parameterization can generalize a specific solution.  5.3.1F Parameters generalize a solution by allowing a function to be used instead of duplicated code  5.3.1G Parameters provide different values as input to procedures when they are called in a program.  5.3.1H Data abstraction provides a means of separating behavior from implementation.  5.3.1I Strings and string operations, including concatenation and some form of substring, are common in many programs.  5.3.1J Integers and floatingpoint numbers are used in programs without requiring understanding of how they are implemented.  5.3.1K Lists and list operations, such as add, remove, and search, are common in many programs.  5.3.1L Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.  5.3.1M Application program interfaces (APIs) and libraries simplify complex programming tasks.  5.3.1N Documentation for an API/library is an important aspect of programming.  5.3.1O APIs connect software components, allowing them to communicate. |
| ~~Define a variable to store a list or string.~~  Represent a list or string using a variable | A list is an ~~indexed sequence of value~~. ordered sequence of elements. For example, [value1, value2, value3, …] where value1 is the first element, value 2 is the second element, value 3 is the third element, …  An element is an individual value in a list which is assigned a unique index.  An index is a common method for ~~keeping track of~~ referencing the elements in a list or string using ~~the~~ natural numbers.  A string is ~~a list of~~ an ordered sequence of characters. |  | 5.3.1 Use abstraction to manage complexity in programs. [P3] | 5.3.1A Procedures are reusable programming abstractions.  5.3.1B A function is a named grouping of programming instructions.  5.3.1C Procedures reduce the complexity of writing and maintaining programs.  5.3.1D Procedures have names and may have parameters and return values.  5.3.1E Parameterization can generalize a specific solution.  5.3.1F Parameters generalize a solution by allowing a function to be used instead of duplicated code  5.3.1G Parameters provide different values as input to procedures when they are called in a program.  5.3.1H Data abstraction provides a means of separating behavior from implementation.  5.3.1I Strings and string operations, including concatenation and some form of substring, are common in many programs.  5.3.1J Integers and floating point numbers are used in programs without requiring understanding of how they are implemented.  5.3.1K Lists and list operations, such as add, remove, and search, are common in many programs.  5.3.1L Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.  5.3.1M Application program interfaces (APIs) and libraries simplify complex programming tasks.  5.3.1N Documentation for an API/library is an important aspect of programming.  5.3.1O APIs connect software components, allowing them to communicate. |
| For data abstraction:  (a) Develop data abstraction using lists or strings to store multiple elements.  (b) Explain how the use of data abstraction manages complexity in a program. | Data abstraction provides a separation between the abstract properties of a data type and the concrete details of its representation.  Data abstractions manage complexity in the program by giving a collection of data a name without the specific details of the representation.  Data abstractions can be created using lists.    ~~Data abstractions can be created using strings by separating items with a comma or other delineating character.~~  Developing a data abstraction to implement in a program can result in a program that is easier to develop and maintain.    Data abstractions ~~can contain heterogeneous elements, using strings or lists~~. often contain different types of elements.  The use of lists allows multiple related items to be represented using a single variable. Lists are referred to by different names, such as array, depending on the programming language.  *Exception*: The use of linked lists is beyond the scope of this course.  The exam reference sheet provides the following for creating a list of values and assigning it to aList, which is a variable that references the new list:   * aList ← [value1, value2, value3...] - assigns aList to a new list containing values for value1, value2, value3, and … at indices 1, 2, 3, and … respectively.   The exam reference sheet describes a list structure whose index values are 1 through the number of elements in the list, inclusive. Using an index value outside of this range will result in an error. |  | 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. [P2]  2.2.2 Use multiple levels of abstraction to write programs. [P3]  5.3.1 Use abstraction to manage complexity in programs. [P3] | 2.2.1A The process of developing an abstraction involves removing detail and generalizing functionality.  2.2.1B An abstraction extracts common features from specific examples in order to generalize concepts.  2.2.1C An abstraction generalizes functionality with input parameters that allow software reuse.  Exclusion Statement: (2.2.1C): An understanding of the difference between value and reference parameters is beyond the scope of this course and the AP Exam.  2.2.2A Software is developed using multiple levels of abstractions, such as constants, expressions, statements, procedures, and libraries.  2.2.2B Being aware of and using multiple levels of abstraction in developing programs helps to more effectively apply available resources and tools to solve problems.  5.3.1A Procedures are reusable programming abstractions.  5.3.1B A function is a named grouping of programming instructions.  5.3.1C Procedures reduce the complexity of writing and maintaining programs.  5.3.1D Procedures have names and may have parameters and return values.  5.3.1E Parameterization can generalize a specific solution.  5.3.1F Parameters generalize a solution by allowing a function to be used instead of duplicated code  5.3.1G Parameters provide different values as input to procedures when they are called in a program.  5.3.1H Data abstraction provides a means of separating behavior from implementation.  5.3.1I Strings and string operations, including concatenation and some form of substring, are common in many programs.  5.3.1J Integers and floatingpoint numbers are used in programs without requiring understanding of how they are implemented.  5.3.1K Lists and list operations, such as add, remove, and search, are common in many programs.  5.3.1L Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.  5.3.1M Application program interfaces (APIs) and libraries simplify complex programming tasks.  5.3.1N Documentation for an API/library is an important aspect of programming.  5.3.1O APIs connect software components, allowing them to communicate. |
| Express an algorithm that uses sequencing without using a programming language. | An algorithm is a finite set of instructions that accomplish a specific task.  Beyond visual and textual programming languages, algorithms can be expressed in a variety of ways, such as natural language, diagrams, and pseudocode.  ~~Natural language, diagrams, and pseudocode describe algorithms without using a programming language.~~  Algorithms executed by a program are implemented using programming languages.  Every algorithm can be constructed using combinations of sequencing, selection, and iteration. |  | 4.1.1 Develop an algorithm for implementation in a program. [P2]  4.1.2 Express an algorithm in a language. [P5] | 4.1.1A Sequencing, selection, and iteration are building blocks of algorithms.  4.1.1B Sequencing is the application of each step of an algorithm in the order in which the statements are given.  4.1.1C Selection uses a Boolean condition to determine which of two parts of an algorithm is used.  4.1.1D Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.  4.1.1E Algorithms can be combined to make new algorithms.  4.1.1F Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.  4.1.1G Knowledge of standard algorithms can help in constructing new algorithms.  4.1.1H Different algorithms can be developed to solve the same problem.  4.1.1I Developing a new algorithm to solve a problem can yield insight into the problem.  4.1.2A Languages for algorithms include natural language, pseudocode, and visual and textual programming languages.  4.1.2B Natural language and pseudocode describe algorithms so that humans can understand them.  4.1.2C Algorithms described in programming languages can be executed on a computer.  4.1.2D Different languages are better suited for expressing different algorithms.  4.1.2E Some programming languages are designed for specific domains and are better for expressing algorithms in those domains.  4.1.2F The language used to express an algorithm can affect characteristics such as clarity or readability but not whether an algorithmic solution exists.  4.1.2G Every algorithm can be constructed using only sequencing, selection, and iteration.  4.1.2H Nearly all programming languages are equivalent in terms of being able to express any algorithm.  4.1.2I Clarity and readability are important considerations when expressing an algorithm in a language. |
| Represent a step-by-step algorithmic process using sequential code statements. | ~~In the software development process, the implementation phase results in an executable program based on the design.~~  Sequencing is the application of each step of an algorithm in the order in which the code statements are given.  A code statement is a part of program code that expresses an action to be carried out.  Expressions can consist of a value, a variable, operators, or procedure calls that return a value.  Expressions are evaluated to produce a single value.  The evaluation of expressions follows a set order of operations defined by the programming language.  Sequential statements execute in the order they appear in the code segment.  Clarity and readability are important considerations when expressing an algorithm in a programming language. |  | 5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1] | 5.5.1A Numbers and numerical concepts are fundamental to programming.  5.5.1B Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.  Exclusion Statement (5.5.1B): Specific range limitations of all programming languages are beyond the scope of this course and the AP Exam.  5.5.1C Real numbers are approximated by floating point representations that do not necessarily have infinite precision.  Exclusion Statement (5.5.1C): Specific sets of values that cannot be exactly represented by floating point numbers are beyond the scope of this course and the AP Exam.  5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.  5.5.1E Logical concepts and Boolean algebra are fundamental to programming.  5.5.1F Compound expressions using and, or, and not are part of most programming languages.  5.5.1G Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.  5.5.1H Computational methods may use lists and collections to solve problems.  5.5.1I Lists and other collections can be treated as abstract data types (ADTs) in developing programs.  5.5.1J Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection. |
| Evaluate expressions that use arithmetic operators. | Arithmetic operators are part of most programming languages and include addition, subtraction, multiplication, division, and modulus operators.  The exam reference sheet provides a MOD b, which evaluates to the remainder when a is divided by b. Assume that a is an integer greater than or equal to 0 and b is an integer greater than 0. For example, 17 MOD 5 evaluates to 2.  The exam reference sheet provides the arithmetic operators +, -, \*, /, and MOD.   * a + b * a - b * a \* b * a / b * a MOD b   These are used to perform arithmetic on a and b. For example, 17 / 5 evaluates to 3.4.  The mathematical order of operations applies when evaluating expressions. |  | 5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1] | 5.5.1A Numbers and numerical concepts are fundamental to programming.  5.5.1B Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.  Exclusion Statement (5.5.1B): Specific range limitations of all programming languages are beyond the scope of this course and the AP Exam.  5.5.1C Real numbers are approximated by floating point representations that do not necessarily have infinite precision.  Exclusion Statement (5.5.1C): Specific sets of values that cannot be exactly represented by floating point numbers are beyond the scope of this course and the AP Exam.  5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.  5.5.1E Logical concepts and Boolean algebra are fundamental to programming.  5.5.1F Compound expressions using and, or, and not are part of most programming languages.  5.5.1G Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.  5.5.1H Computational methods may use lists and collections to solve problems.  5.5.1I Lists and other collections can be treated as abstract data types (ADTs) in developing programs.  5.5.1J Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection. |
| Evaluate expressions that manipulate strings. | String concatenation joins together two or more strings end-to-end to make a new string.  A substring is part of an existing string. |  | 5.3.1 Use abstraction to manage complexity in programs. [P3] | 5.3.1A Procedures are reusable programming abstractions.  5.3.1B A function is a named grouping of programming instructions.  5.3.1C Procedures reduce the complexity of writing and maintaining programs.  5.3.1D Procedures have names and may have parameters and return values.  5.3.1E Parameterization can generalize a specific solution.  5.3.1F Parameters generalize a solution by allowing a function to be used instead of duplicated code  5.3.1G Parameters provide different values as input to procedures when they are called in a program.  5.3.1H Data abstraction provides a means of separating behavior from implementation.  5.3.1I Strings and string operations, including concatenation and some form of substring, are common in many programs.  5.3.1J Integers and floatingpoint numbers are used in programs without requiring understanding of how they are implemented.  5.3.1K Lists and list operations, such as add, remove, and search, are common in many programs.  5.3.1L Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.  5.3.1M Application program interfaces (APIs) and libraries simplify complex programming tasks.  5.3.1N Documentation for an API/library is an important aspect of programming.  5.3.1O APIs connect software components, allowing them to communicate. |
| For relationships between two variables, expressions, or values:  (a) Represent using relational operators  (b) Evaluate expressions that use relational operators | A Boolean value is either true or false.  The exam reference sheet provides the following relational operators =, ≠, >, <, ≥, and ≤.   * a = b * a ≠ b * a > b * a < b * a ≥ b * a ≤ b   These are used to test the relationship between two variables, expressions, or values and evaluate to a Boolean value.  For example, a = b evaluates to true if a and b are equal; otherwise, it evaluates to false. |  | 5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1] | 5.5.1A Numbers and numerical concepts are fundamental to programming.  5.5.1B Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.  Exclusion Statement (5.5.1B): Specific range limitations of all programming languages are beyond the scope of this course and the AP Exam.  5.5.1C Real numbers are approximated by floating point representations that do not necessarily have infinite precision.  Exclusion Statement (5.5.1C): Specific sets of values that cannot be exactly represented by floating point numbers are beyond the scope of this course and the AP Exam.  5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.  5.5.1E Logical concepts and Boolean algebra are fundamental to programming.  5.5.1F Compound expressions using and, or, and not are part of most programming languages.  5.5.1G Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.  5.5.1H Computational methods may use lists and collections to solve problems.  5.5.1I Lists and other collections can be treated as abstract data types (ADTs) in developing programs.  5.5.1J Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection. |
| For relationships between Boolean values:  (a) Represent using logical operators.  (b) Evaluate expressions that use logic operators. | The exam reference sheet provides the following logic operators: NOT, AND, and OR, which evaluate to a Boolean value.  The exam reference sheet provides NOT condition, which evaluates to true if condition is false; otherwise it evaluates to false.  The exam reference sheet provides condition1 AND condition2, which evaluates to true if both condition1 and condition2 are true; otherwise it evaluates to false.  The exam reference sheet provides condition1 OR condition2, which evaluates to true if condition1 is true or if condition2 is true or if both condition1 and condition2 are true; otherwise it evaluates to false.  The operands for a logic operator are either a Boolean expression or a Boolean value. |  | 5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1] | 5.5.1A Numbers and numerical concepts are fundamental to programming.  5.5.1B Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.  Exclusion Statement (5.5.1B): Specific range limitations of all programming languages are beyond the scope of this course and the AP Exam.  5.5.1C Real numbers are approximated by floating point representations that do not necessarily have infinite precision.  Exclusion Statement (5.5.1C): Specific sets of values that cannot be exactly represented by floating point numbers are beyond the scope of this course and the AP Exam.  5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.  5.5.1E Logical concepts and Boolean algebra are fundamental to programming.  5.5.1F Compound expressions using and, or, and not are part of most programming languages.  5.5.1G Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.  5.5.1H Computational methods may use lists and collections to solve problems.  5.5.1I Lists and other collections can be treated as abstract data types (ADTs) in developing programs.  5.5.1J Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection. |
| Express an algorithm that uses selection without using a programming language. | Selection determines which parts of an algorithm are executed based on a condition being true or false. |  | 4.1.1 Develop an algorithm for implementation in a program. [P2] | 4.1.1A Sequencing, selection, and iteration are building blocks of algorithms.  4.1.1B Sequencing is the application of each step of an algorithm in the order in which the statements are given.  4.1.1C Selection uses a Boolean condition to determine which of two parts of an algorithm is used.  4.1.1D Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.  4.1.1E Algorithms can be combined to make new algorithms.  4.1.1F Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.  4.1.1G Knowledge of standard algorithms can help in constructing new algorithms.  4.1.1H Different algorithms can be developed to solve the same problem.  4.1.1I Developing a new algorithm to solve a problem can yield insight into the problem. |
| For a selection process:  (a) Represent using conditional statements (b) Determine the result of conditional statements | Conditional statements or “if-statements” affect the sequential flow of control by executing different statements based on the value of a Boolean expression.  The exam reference sheet provides  IF(condition)  {  <block of statements>  }  in which the code in block of statements is executed if the Boolean expression condition evaluates to true; no action is taken if condition evaluates to false.  The exam reference sheet provides IF(condition) {  <first block of statements> } ELSE {  <second block of statements> } in which the code in first block of statements is executed if the Boolean expression condition evaluates to true; otherwise, the code in second block of statements is executed. |  | 5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1] | 5.5.1A Numbers and numerical concepts are fundamental to programming.  5.5.1B Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.  Exclusion Statement (5.5.1B): Specific range limitations of all programming languages are beyond the scope of this course and the AP Exam.  5.5.1C Real numbers are approximated by floating point representations that do not necessarily have infinite precision.  Exclusion Statement (5.5.1C): Specific sets of values that cannot be exactly represented by floating point numbers are beyond the scope of this course and the AP Exam.  5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.  5.5.1E Logical concepts and Boolean algebra are fundamental to programming.  5.5.1F Compound expressions using and, or, and not are part of most programming languages.  5.5.1G Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.  5.5.1H Computational methods may use lists and collections to solve problems.  5.5.1I Lists and other collections can be treated as abstract data types (ADTs) in developing programs.  5.5.1J Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection. |
| For nested selection processes:  (a)Represent using nested conditional statements.  (b) Determine the result of nested conditional statements. | Nested If-statements or “else if” statements consist of If-statements within If-statements.  If the Boolean condition of the initial if evaluates to false, then the Boolean condition of the ~~“else if” is~~ nested conditional statement is evaluated. |  | 5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1] | 5.5.1A Numbers and numerical concepts are fundamental to programming.  5.5.1B Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.  Exclusion Statement (5.5.1B): Specific range limitations of all programming languages are beyond the scope of this course and the AP Exam.  5.5.1C Real numbers are approximated by floating point representations that do not necessarily have infinite precision.  Exclusion Statement (5.5.1C): Specific sets of values that cannot be exactly represented by floating point numbers are beyond the scope of this course and the AP Exam.  5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.  5.5.1E Logical concepts and Boolean algebra are fundamental to programming.  5.5.1F Compound expressions using and, or, and not are part of most programming languages.  5.5.1G Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.  5.5.1H Computational methods may use lists and collections to solve problems.  5.5.1I Lists and other collections can be treated as abstract data types (ADTs) in developing programs.  5.5.1J Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection. |
| Express an algorithm that uses iteration without using a programming language. | Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times. |  | 4.1.1 Develop an algorithm for implementation in a program. [P2] | 4.1.1A Sequencing, selection, and iteration are building blocks of algorithms.  4.1.1B Sequencing is the application of each step of an algorithm in the order in which the statements are given.  4.1.1C Selection uses a Boolean condition to determine which of two parts of an algorithm is used.  4.1.1D Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.  4.1.1E Algorithms can be combined to make new algorithms.  4.1.1F Using existing correct algorithms as building blocks for constructing a new algorithm helps ensure the new algorithm is correct.  4.1.1G Knowledge of standard algorithms can help in constructing new algorithms.  4.1.1H Different algorithms can be developed to solve the same problem.  4.1.1I Developing a new algorithm to solve a problem can yield insight into the problem. |
| For iterative processes:  (a) Represent using iteration statements.  (b) Determine the result or side-effect of iteration statements. | Iteration statements change the sequential flow of control by repeating a set of statements zero or more times until a condition is met.  The exam reference sheet provides REPEAT n TIMES {  <block of statements> }  in which the block of statements is executed n times.  The exam reference sheet provides REPEAT UNTIL(condition) {  <block of statements> } in which the code in block of statements is repeated until the Boolean expression condition evaluates to true.  In REPEAT UNTIL (condition)iteration, an infinite loop occurs when the condition will never evaluate to true.  In REPEAT UNTIL (condition)iteration, if the conditional evaluates to true initially, the loop body is not executed at all due to the condition being checked before the loop. |  | 5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1] | 5.5.1A Numbers and numerical concepts are fundamental to programming.  5.5.1B Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.  Exclusion Statement (5.5.1B): Specific range limitations of all programming languages are beyond the scope of this course and the AP Exam.  5.5.1C Real numbers are approximated by floating point representations that do not necessarily have infinite precision.  Exclusion Statement (5.5.1C): Specific sets of values that cannot be exactly represented by floating point numbers are beyond the scope of this course and the AP Exam.  5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.  5.5.1E Logical concepts and Boolean algebra are fundamental to programming.  5.5.1F Compound expressions using and, or, and not are part of most programming languages.  5.5.1G Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.  5.5.1H Computational methods may use lists and collections to solve problems.  5.5.1I Lists and other collections can be treated as abstract data types (ADTs) in developing programs.  5.5.1J Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection. |
| Compare multiple ~~equivalent representations of a process.~~ algorithms to determine if they yield the same side effect or result. | Algorithms can be written in different ways and still accomplish the same tasks.  Algorithms that appear similar can yield different side-effects or results.  ~~Code segments can be written in different ways and still accomplish the same tasks.~~  Some selection processes can be written as equivalent Boolean expressions.  Some Boolean expressions can be written as equivalent selection processes.  Different algorithms can be developed or used to solve the same problem. |  | 5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1] | 5.5.1A Numbers and numerical concepts are fundamental to programming.  5.5.1B Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.  Exclusion Statement (5.5.1B): Specific range limitations of all programming languages are beyond the scope of this course and the AP Exam.  5.5.1C Real numbers are approximated by floating point representations that do not necessarily have infinite precision.  Exclusion Statement (5.5.1C): Specific sets of values that cannot be exactly represented by floating point numbers are beyond the scope of this course and the AP Exam.  5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.  5.5.1E Logical concepts and Boolean algebra are fundamental to programming.  5.5.1F Compound expressions using and, or, and not are part of most programming languages.  5.5.1G Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.  5.5.1H Computational methods may use lists and collections to solve problems.  5.5.1I Lists and other collections can be treated as abstract data types (ADTs) in developing programs.  5.5.1J Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection. |
| ~~Combine and modify existing algorithms to create a new algorithm.~~  For algorithms:  (a) Create algorithms.  (b) Combine and modify existing algorithms. | ~~Algorithms can be combined to make new algorithms.~~Algorithms can be created from an idea, by combining existing algorithms, or by modifying existing algorithms.  Knowledge of existing algorithms can help in constructing new algorithms. Some existing algorithms include:   * determining the maximum or minimum value of 2 or more numbers; * computing the sum or average of 2 or more numbers; * identifying if an integer is or is not evenly divisible by another integer; * determining a robot’s path through a maze.   Using existing tested algorithms as building blocks for constructing a new algorithm has benefits such as reducing development time, reducing testing, and simplifying the identification of errors.  ~~Different algorithms can be developed or used to solve the same problem.~~ |  | 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. [P2] | 1.2.3A Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.  1.2.3B Computation facilitates the creation and modification of computational artifacts with enhanced detail and precision.  1.2.3C Combining or modifying existing artifacts can show personal expression of ideas. |
| For list operations:  (a) Write expressions that use list indexing and list procedures.  (b) Evaluate expressions that use list indexing and list procedures. | ~~Basic operations on lists include:~~   * ~~accessing an element by index,~~ * ~~inserting elements at a given index,~~ * ~~adding elements to the end of the list (append),~~ * ~~removing elements, and~~ * ~~determining the length of a list.~~   ~~These are implemented in accordance with the syntax rules of the language.~~  ~~The exam reference sheet provides the following for list aList:~~   * ~~aList[i] - refer to the element of aList at index i. The first element of aList is at index 1.~~ * ~~aList[i] ← aList[j] - assigns the value of aList[j] to aList[i].~~ * ~~x ← aList [i] - assigns the value of aList[i] to the variable x.~~ * ~~aList[i] ← x - assigns the value of x to aList[i].~~ * ~~INSERT(aList, i, value) - any variables in aList at indices greater than or equal to i are shifted one location to the right. The length of the list is increased by 1, and value is placed at index i in aList.~~ * ~~APPEND(aList, value) - the length of aList is increased by 1, and value is placed at the end of the aList.~~ * ~~REMOVE(aList, i) - Removes the item at index i in aList and shifts to the left any values at indices greater than i. The length of aList is decreased by 1.~~ * ~~LENGTH(aList) – Evaluates to the number of elements in aList.~~ * Basic operations on lists include:   + accessing an element by index;     - aList[i] refers to the element of aList at index i. The first element of aList is at index 1, alist[1].   + assigning a value of an element of a list to a variable;     - x ← aList [i] - assigns the value of aList[i] to the variable x.inserting elements at a given index,   + assigning a value to an element of a list;     - aList[i] ← x - assigns the value of x to aList[i].     - aList[i] ← aList[j] - assigns the value of aList[j] to aList[i].   + inserting elements at a given index;     - INSERT(aList, i, value) - any variables in aList at indices greater than or equal to i are shifted one location to the right. The length of the list is increased by 1, and value is placed at index i in aList.   + adding elements to the end of the list (append);     - APPEND(aList, value) - the length of aList is increased by 1, and value is placed at the end of the aList.   + removing elements; and     - REMOVE(aList, i) - Removes the item at index i in aList and shifts to the left any values at indices greater than i. The length of aList is decreased by 1.   + determining the length of a list.     - LENGTH(aList) – Evaluates to the number of elements currently in aList. * List procedures are implemented in accordance with the syntax rules of the language. | These changes pair the operations with what students will see in the exam reference. | 5.3.1 Use abstraction to manage complexity in programs. [P3]  5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1] | 5.3.1A Procedures are reusable programming abstractions.  5.3.1B A function is a named grouping of programming instructions.  5.3.1C Procedures reduce the complexity of writing and maintaining programs.  5.3.1D Procedures have names and may have parameters and return values.  5.3.1E Parameterization can generalize a specific solution.  5.3.1F Parameters generalize a solution by allowing a function to be used instead of duplicated code  5.3.1G Parameters provide different values as input to procedures when they are called in a program.  5.3.1H Data abstraction provides a means of separating behavior from implementation.  5.3.1I Strings and string operations, including concatenation and some form of substring, are common in many programs.  5.3.1J Integers and floatingpoint numbers are used in programs without requiring understanding of how they are implemented.  5.3.1K Lists and list operations, such as add, remove, and search, are common in many programs.  5.3.1L Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.  5.3.1M Application program interfaces (APIs) and libraries simplify complex programming tasks.  5.3.1N Documentation for an API/library is an important aspect of programming.  5.3.1O APIs connect software components, allowing them to communicate.  5.5.1A Numbers and numerical concepts are fundamental to programming.  5.5.1B Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.  Exclusion Statement (5.5.1B): Specific range limitations of all programming languages are beyond the scope of this course and the AP Exam.  5.5.1C Real numbers are approximated by floating point representations that do not necessarily have infinite precision.  Exclusion Statement (5.5.1C): Specific sets of values that cannot be exactly represented by floating point numbers are beyond the scope of this course and the AP Exam.  5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.  5.5.1E Logical concepts and Boolean algebra are fundamental to programming.  5.5.1F Compound expressions using and, or, and not are part of most programming languages.  5.5.1G Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.  5.5.1H Computational methods may use lists and collections to solve problems.  5.5.1I Lists and other collections can be treated as abstract data types (ADTs) in developing programs.  5.5.1J Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection. |
| For ~~traversals of a list~~ algorithms involving elements of a list:  (a) Represent using iterative statements to traverse lists.  (b) Determine the result of an algorithm with list traversals. | Traversing a list can be a complete traversal where all elements in the list are accessed, or a partial traversal where only a portion of elements are accessed. Exclusion Statement: Traversing multiple lists at the same time using the same index for both, parallel traversals, is beyond the scope of this course.  Iteration statements can be used to ~~access all the elements in~~ traverse a list.  The exam reference sheet provides  FOR EACH item IN aList {  <block of statements> }  The variable item is assigned the value of each element of aList sequentially, in order from the first element to the last element. The code in block of statements is executed once for each assignment of item.  Knowledge of existing algorithms that use iteration can help in constructing new algorithms. Some examples of existing algorithms that are often used with lists include:   * determining a minimum or maximum value in a list. * computing a sum or average of a list of numbers.   ~~Some algorithms require multiple lists to be traversed simultaneously.~~  ~~Sequential / linear~~ Linear search or sequential search algorithms check each element of a list in order until the desired value is found or all elements in the list have been checked. ~~start at the beginning of the list and check each element in order until the desired value is found or the end of the list is reached.~~ | Removed LO on multiple simultaneous traversals of a list and added this exclusion. This also reduces the scope of the current. | 5.3.1 Use abstraction to manage complexity in programs. [P3]  5.5.1 Employ appropriate mathematical and logical concepts in programming. [P1] | 5.3.1A Procedures are reusable programming abstractions.  5.3.1B A function is a named grouping of programming instructions.  5.3.1C Procedures reduce the complexity of writing and maintaining programs.  5.3.1D Procedures have names and may have parameters and return values.  5.3.1E Parameterization can generalize a specific solution.  5.3.1F Parameters generalize a solution by allowing a function to be used instead of duplicated code  5.3.1G Parameters provide different values as input to procedures when they are called in a program.  5.3.1H Data abstraction provides a means of separating behavior from implementation.  5.3.1I Strings and string operations, including concatenation and some form of substring, are common in many programs.  5.3.1J Integers and floatingpoint numbers are used in programs without requiring understanding of how they are implemented.  5.3.1K Lists and list operations, such as add, remove, and search, are common in many programs.  5.3.1L Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.  5.3.1M Application program interfaces (APIs) and libraries simplify complex programming tasks.  5.3.1N Documentation for an API/library is an important aspect of programming.  5.3.1O APIs connect software components, allowing them to communicate.  5.5.1A Numbers and numerical concepts are fundamental to programming.  5.5.1B Integers may be constrained in the maximum and minimum values that can be represented in a program because of storage limitations.  Exclusion Statement (5.5.1B): Specific range limitations of all programming languages are beyond the scope of this course and the AP Exam.  5.5.1C Real numbers are approximated by floating point representations that do not necessarily have infinite precision.  Exclusion Statement (5.5.1C): Specific sets of values that cannot be exactly represented by floating point numbers are beyond the scope of this course and the AP Exam.  5.5.1D Mathematical expressions using arithmetic operators are part of most programming languages.  5.5.1E Logical concepts and Boolean algebra are fundamental to programming.  5.5.1F Compound expressions using and, or, and not are part of most programming languages.  5.5.1G Intuitive and formal reasoning about program components using Boolean concepts helps in developing correct programs.  5.5.1H Computational methods may use lists and collections to solve problems.  5.5.1I Lists and other collections can be treated as abstract data types (ADTs) in developing programs.  5.5.1J Basic operations on collections include adding elements, removing elements, iterating over all elements, and determining whether an element is in a collection. |
| For binary search algorithms:  (a) Determine the number of iterations required to find a value in a data set.  (b) Explain the requirements necessary to complete a binary search. | The binary search algorithm starts at the middle of a sorted data set of numbers and eliminates half of the data in each iteration until the desired value is found or all elements have been eliminated.  Exclusion: Specific implementations of the binary search are not tested on the exam.  Data must be in sorted order to use the binary search algorithm.  Binary search is often more efficient than sequential / linear search when the data is sorted. |  | 4.2.4 Evaluate algorithms analytically and empirically for efficiency, correctness, and clarity. [P4] | 4.2.4A Determining an algorithm's efficiency is done by reasoning formally or mathematically about the algorithm.  4.2.4B Empirical analysis of an algorithm is done by implementing the algorithm and running it on different inputs.  4.2.4C The correctness of an algorithm is determined by reasoning formally or mathematically about the algorithm, not by testing an implementation of the algorithm.  Exclusion Statement (4.2.4C): Formally proving program correctness is beyond the scope of this course and the AP Exam.  4.2.4D Different correct algorithms for the same problem can have different efficiencies.  4.2.4E Sometimes more efficient algorithms are more complex.  4.2.4F Finding an efficient algorithm for a problem can help solve larger instances of the problem.  4.2.4G Efficiency includes both execution time and memory usage.  Exclusion Statement (4.2.4G): Formal analysis of algorithms (BigO) and formal reasoning using mathematical formulas are beyond the scope of this course and the AP Exam.  4.2.4H Linear search can be used when searching for an item in any list; binary search can be used only when the list is sorted. |
| ~~Determine the effect of using display.~~ | ~~The exam reference sheet provides DISPLAY(expression) to display the value of expression, followed by a space.~~ | EK statement moved to another LO. | 5.3.1 Use abstraction to manage complexity in programs. [P3] | 5.3.1A Procedures are reusable programming abstractions.  5.3.1B A function is a named grouping of programming instructions.  5.3.1C Procedures reduce the complexity of writing and maintaining programs.  5.3.1D Procedures have names and may have parameters and return values.  5.3.1E Parameterization can generalize a specific solution.  5.3.1F Parameters generalize a solution by allowing a function to be used instead of duplicated code  5.3.1G Parameters provide different values as input to procedures when they are called in a program.  5.3.1H Data abstraction provides a means of separating behavior from implementation.  5.3.1I Strings and string operations, including concatenation and some form of substring, are common in many programs.  5.3.1J Integers and floatingpoint numbers are used in programs without requiring understanding of how they are implemented.  5.3.1K Lists and list operations, such as add, remove, and search, are common in many programs.  5.3.1L Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.  5.3.1M Application program interfaces (APIs) and libraries simplify complex programming tasks.  5.3.1N Documentation for an API/library is an important aspect of programming.  5.3.1O APIs connect software components, allowing them to communicate. |
| ~~Determine the effect of code segments that accept input from a user.~~ | ~~The exam reference sheet provides INPUT(), which accepts a value from the user and it is equivalent to the entered value.~~ | EK statement moved to another LO. | 5.3.1 Use abstraction to manage complexity in programs. [P3] | 5.3.1A Procedures are reusable programming abstractions.  5.3.1B A function is a named grouping of programming instructions.  5.3.1C Procedures reduce the complexity of writing and maintaining programs.  5.3.1D Procedures have names and may have parameters and return values.  5.3.1E Parameterization can generalize a specific solution.  5.3.1F Parameters generalize a solution by allowing a function to be used instead of duplicated code  5.3.1G Parameters provide different values as input to procedures when they are called in a program.  5.3.1H Data abstraction provides a means of separating behavior from implementation.  5.3.1I Strings and string operations, including concatenation and some form of substring, are common in many programs.  5.3.1J Integers and floatingpoint numbers are used in programs without requiring understanding of how they are implemented.  5.3.1K Lists and list operations, such as add, remove, and search, are common in many programs.  5.3.1L Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.  5.3.1M Application program interfaces (APIs) and libraries simplify complex programming tasks.  5.3.1N Documentation for an API/library is an important aspect of programming.  5.3.1O APIs connect software components, allowing them to communicate. |
| Determine the result or effect of a procedure call. | A procedure is a named group of programming instructions and may have parameters and return values.  Procedures are referred to by different names, such as method or function, depending on the programming language.  Parameters are input variables of a procedure. Arguments specify the values of the parameters when a procedure is called.  A procedure call interrupts the sequential execution of statements, causing the program to first execute the statements within the procedure before continuing. Once the last statement in the procedure has executed or a return statement is executed, flow of control is returned to the point immediately following where the procedure was called.  The exam reference sheet provides  NameOfProcedure (value1, value2, …) as a way to call PROCEDURE NameOfProcedure (parameter1, parameter 2…) which takes zero or more parameters and value1 is assigned to parameter1 and value2 is assigned to parameter2.  The exam reference sheet provides the procedure DISPLAY(expression) to display the value of expression, followed by a space.  In the exam reference sheet, the RETURN keyword, which is used to return the flow of control to the point where the procedure was called and to return a value.  The exam reference sheet provides procedure INPUT(), which accepts a value from the user and is equivalent to the input value.  ~~When a value is returned the procedure call is replaced with the value of expression.~~  The “value of a procedure” is the value returned from a call to the procedure. | Display and input where added here… in red | 5.3.1 Use abstraction to manage complexity in programs. [P3] | 5.3.1A Procedures are reusable programming abstractions.  5.3.1B A function is a named grouping of programming instructions.  5.3.1C Procedures reduce the complexity of writing and maintaining programs.  5.3.1D Procedures have names and may have parameters and return values.  5.3.1E Parameterization can generalize a specific solution.  5.3.1F Parameters generalize a solution by allowing a function to be used instead of duplicated code  5.3.1G Parameters provide different values as input to procedures when they are called in a program.  5.3.1H Data abstraction provides a means of separating behavior from implementation.  5.3.1I Strings and string operations, including concatenation and some form of substring, are common in many programs.  5.3.1J Integers and floatingpoint numbers are used in programs without requiring understanding of how they are implemented.  5.3.1K Lists and list operations, such as add, remove, and search, are common in many programs.  5.3.1L Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.  5.3.1M Application program interfaces (APIs) and libraries simplify complex programming tasks.  5.3.1N Documentation for an API/library is an important aspect of programming.  5.3.1O APIs connect software components, allowing them to communicate. |
| Explain how the use of procedural abstraction manages complexity in a program. | One common type of abstraction is procedural abstraction which provides a name for a process and allows a procedure to be used only knowing what it does, not how it does it.  Procedural abstraction allows a solution to a large problem to be based on the solution of smaller sub-problems. This is accomplished by creating procedures to solve each one of the sub-problems.  ~~Developing correct procedures and then combining them helps in creating correct programs. This is called modularity.~~  ~~Incrementally adding tested code segments to a correct program helps create larger correct programs.~~  The process of subdividing a computer program into separate sub-programs is called modularity.  A procedural abstraction may extract shared features to generalize functionality ~~through the use of input parameters~~ instead of duplicating code. This allows for program code reuse, which helps to manage complexity.  Using procedural abstraction helps improve code readability.  Using procedural abstraction in a program allows programmers to change the internals of the procedure (to make it faster, more efficient, use less storage, etc.) without needing to notify users of the change as long as what the procedure does is preserved. | This LO was split into 2 LOs. | 2.2.1 Develop an abstraction when writing a program or creating other computational artifacts. [P2]  2.2.2 Use multiple levels of abstraction to write programs. [P3]  5.3.1 Use abstraction to manage complexity in programs. [P3] | 2.2.1A The process of developing an abstraction involves removing detail and generalizing functionality.  2.2.1B An abstraction extracts common features from specific examples in order to generalize concepts.  2.2.1C An abstraction generalizes functionality with input parameters that allow software reuse.  Exclusion Statement: (2.2.1C): An understanding of the difference between value and reference parameters is beyond the scope of this course and the AP Exam.  2.2.2A Software is developed using multiple levels of abstractions, such as constants, expressions, statements, procedures, and libraries.  2.2.2B Being aware of and using multiple levels of abstraction in developing programs helps to more effectively apply available resources and tools to solve problems.  5.3.1A Procedures are reusable programming abstractions.  5.3.1B A function is a named grouping of programming instructions.  5.3.1C Procedures reduce the complexity of writing and maintaining programs.  5.3.1D Procedures have names and may have parameters and return values.  5.3.1E Parameterization can generalize a specific solution.  5.3.1F Parameters generalize a solution by allowing a function to be used instead of duplicated code  5.3.1G Parameters provide different values as input to procedures when they are called in a program.  5.3.1H Data abstraction provides a means of separating behavior from implementation.  5.3.1I Strings and string operations, including concatenation and some form of substring, are common in many programs.  5.3.1J Integers and floatingpoint numbers are used in programs without requiring understanding of how they are implemented.  5.3.1K Lists and list operations, such as add, remove, and search, are common in many programs.  5.3.1L Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.  5.3.1M Application program interfaces (APIs) and libraries simplify complex programming tasks.  5.3.1N Documentation for an API/library is an important aspect of programming.  5.3.1O APIs connect software components, allowing them to communicate. |
| ~~For procedural abstraction:~~  ~~(a) Develop procedural abstraction to manage complexity in a program by writing procedures. (b) Explain how the use of procedural abstraction manages~~ ~~complexity in a program.~~  Develop procedural abstractions to manage complexity in a program by writing procedures. | The exam reference sheet provides  PROCEDURE NameOfProcedure (parameter1, parameter2, …)  {  <block of statements>  }  ~~A procedure, NameOfProcedure, takes zero or more parameters. The procedure contains the programming instructions notated by <block of statements>.~~that is used to define a procedure that takes zero or more arguments.    The exam reference sheet provides  PROCEDURE NameOfProcedure (parameter1, parameter2, …)  {  <block of statements>  RETURN (expression)  }  ~~A procedure, NameOfProcedure, takes zero or more parameters. The procedure contains programming instructions and returns the value of expression.~~ ~~The RETURN statement may appear at any point inside the procedure and causes an immediate return from the procedure back to where the procedure was called, replacing the procedure call with the value of expression~~.  that is used to define a procedure that takes zero or more arguments and contains programming instructions and returns the value of expression. |
| Select appropriate libraries or existing code segments to use in creating new programs. | Software libraries contain procedures that may be helpful in creating new programs.  Existing code segments can come from internal or external sources, such as:   * libraries; and * previously written code.   The use of libraries simplifies the task of creating complex programs.  Application program interfaces (APIs) are specifications for how the procedures in a library behave and can be used.  Documentation for an API/library is necessary in understanding the behavior(s) provided by the API/library and how to use them. |  | 1.2.3 Create a new computational artifact by combining or modifying existing artifacts. [P2]  5.3.1 Use abstraction to manage complexity in programs. [P3] | 1.2.3A Creating computational artifacts can be done by combining and modifying existing artifacts or by creating new artifacts.  1.2.3B Computation facilitates the creation and modification of computational artifacts with enhanced detail and precision.  1.2.3C Combining or modifying existing artifacts can show personal expression of ideas.  5.3.1A Procedures are reusable programming abstractions.  5.3.1B A function is a named grouping of programming instructions.  5.3.1C Procedures reduce the complexity of writing and maintaining programs.  5.3.1D Procedures have names and may have parameters and return values.  5.3.1E Parameterization can generalize a specific solution.  5.3.1F Parameters generalize a solution by allowing a function to be used instead of duplicated code  5.3.1G Parameters provide different values as input to procedures when they are called in a program.  5.3.1H Data abstraction provides a means of separating behavior from implementation.  5.3.1I Strings and string operations, including concatenation and some form of substring, are common in many programs.  5.3.1J Integers and floatingpoint numbers are used in programs without requiring understanding of how they are implemented.  5.3.1K Lists and list operations, such as add, remove, and search, are common in many programs.  5.3.1L Using lists and procedures as abstractions in programming can result in programs that are easier to develop and maintain.  5.3.1M Application program interfaces (APIs) and libraries simplify complex programming tasks.  5.3.1N Documentation for an API/library is an important aspect of programming.  5.3.1O APIs connect software components, allowing them to communicate. |
| For generating random values:  (a) Write expressions to generate options.  (b)Evaluate expressions to determine the options that may be executed. | The exam reference sheet provides RANDOM(a, b) which represents the generation of a random integer from a to b, including a and b. For example, RANDOM(1, 3) could evaluate to 1, 2, or 3.  Using random number generation in a program means each execution could produce a unique result. Therefore, the behavior needs to be abstracted and generalized for all cases. | This is not considered new as it is on the exam reference sheet.  Reminder – all the content from the exam reference sheet has been included in the curriculum framework to better signal to teachers that this material is required. This contributes to many additional LOs. This should be viewed as greater transparency to teachers, rather than a growth in scope! | **NEW — no 2018 comparison**  **\*\*\* This exists in the exam reference sheet and is required knowledge for students. Crystal** | |
| For simulations:  (a) Explain how computers can be used to represent real-world phenomena or outcomes.  (b) Compare the use of simulations with real-world contexts. | * Simulations are abstractions of more complex objects or phenomena for a specific purpose. * A simulation ~~focuses on the state of a phenomenon for varying sets of values for the variables.~~ is a representation that uses varying sets of values to reflect the changing state of the phenomenon. * Simulations often mimic real-world events with the purpose of drawing inferences allowing investigation of the phenomenon without the constraints of the real-world. * The process of developing an abstract simulation involves removing specific details or simplifying functionality. * All simulations contain bias derived from the choices of elements of the real world that were included or excluded. * Simulations are most useful when real-world events are impractical for experiments (i.e., too big, too small, too fast, too slow, too expensive or too dangerous). * Simulations facilitate the formulation and refinement of hypotheses related to the objects or phenomena under consideration. * ~~Hypotheses are formulated to explain the objects or phenomena being represented.~~ * ~~Hypotheses are refined by examining the insights that simulations provide into the objects or phenomena.~~ * ~~The results of simulations may generate new knowledge and new hypotheses related to the phenomena being represented.~~ * ~~Simulations allow hypotheses to be tested without the constraints of the real world.~~ * Random number generators can be used to simulate the variability that exists in the real-world. | Reduced scope and redundancy in this section. | 2.3.1 Use models and simulations to represent phenomena. [P3]  2.3.2 Use models and simulations to formulate, refine, and test hypotheses. [P3] | 2.3.1A Models and simulations are simplified representations of more complex objects or phenomena.  2.3.1B Models may use different abstractions or levels of abstraction depending on the objects or phenomena being posed.  2.3.1C Models often omit unnecessary features of the objects or phenomena that are being modeled.  2.3.1D Simulations mimic real world events without the cost or danger of building and testing the phenomena in the real world.  2.3.2A Models and simulations facilitate the formulation and refinement of hypotheses related to the objects or phenomena under consideration.  2.3.2B Hypotheses are formulated to explain the objects or phenomena being modeled.  2.3.2C Hypotheses are refined by examining the insights that models and simulations provide into the objects or phenomena.  2.3.2D The results of simulations may generate new knowledge and new hypotheses related to the phenomena being modeled.  2.3.2E Simulations allow hypotheses to be tested without the constraints of the real world.  2.3.2F Simulations can facilitate extensive and rapid testing of models.  2.3.2G The time required for simulations is impacted by the level of detail and quality of the models, and the software and hardware used for the simulation.  2.3.2H Rapid and extensive testing allows models to be changed to accurately reflect the objects or phenomena being modeled. |

Big Idea 4: Computing Systems and Networks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2020 | | Rationale | | 2018 | |
| Learning Objectives | Essential Knowledge |  | | Learning Objectives | Essential Knowledge |
| Students must be able to… | Students will know that… |  | | Students must be able to… | Students will know that… |
| Explain how computing devices work together in a network. | * A computing device is a physical artifact that can run a program. Some examples include computers, tablets, servers, routers, and smart sensors. * A computing system is a group of computing devices and programs working together for a common purpose. * A computer network is a group of interconnected computing devices capable of sending or receiving data.~~sharing data.~~ * A computer network is a type~~an example~~ of a computing system. * A path between two computing devices on a computer network (a sender or a receiver) is a sequence of directly-connected computing devices beginning at the sender and ending at the receiver. * Routing is the process of finding a path from sender to receiver. * The bandwidth of a computer network is the maximum amount of data that can be sent in a fixed amount of time. * Bandwidth is usually measured in bits per second. | Eks in red were from an LO that was removed.  More of an emphasis on the Internet being the computer system of study per provider feedback. | | 6.1.1 Explain the abstractions in the Internet and how the Internet functions. [P3] | Exclusion Statement (LO 6.1.1): Specific devices used to implement the abstractions in the Internet are beyond the scope of this course and the AP Exam.  6.1.1A The Internet connects devices and networks all over the world.  6.1.1B An end to end architecture facilitates connecting new devices and networks on the Internet.  6.1.1C Devices and networks that make up the Internet are connected and communicate using addresses and protocols.  6.1.1D The Internet and the systems built on it facilitate collaboration.  6.1.1E Connecting new devices to the Internet is enabled by assignment of an Internet protocol (IP) address.  6.1.1F The Internet is built on evolving standards, including those for addresses and names.  Exclusion Statement (6.1.1F): Specific details of any particular standard for addresses are beyond the scope of this course and the AP Exam.  6.1.1G The domain name system (DNS) translates domain names to IP addresses.  6.1.1H The number of devices that could use an IP address has grown so fast that a new protocol (IPv6) has been established to handle routing of many more devices.  6.1.1I Standards such as hypertext transfer protocol (HTTP), IP, and simple mail transfer protocol (SMTP) are developed and overseen by the Internet Engineering Task Force (IETF). |
| Explain how the Internet works | * The Internet is a computer network consisting of interconnected networks that use standardized, open or non-proprietary, communication protocols. * ~~The Internet spans the globe and beyond, reaching into space.~~Access to the Internet depends on the ability to connect a computing device to a part of the Internet. * A protocol is an agreed-upon set of rules that specify the behavior of some system. * The protocols used in the Internet are open which allows users to easily connect additional computing devices to the Internet. * Routing on the internet is usually dynamic; it is not specified in advance. * Scalability of a system is the capacity for the system to change in size and scale to meet new demands. * ~~Scalability of systems is an important consideration when working with large data sets, since the size or scale of a system affects how that data set can be processed and stored.~~ * The Internet was designed to be scalable. |
| ~~Describe the bandwidth of a computing network connection.~~ | ~~The bandwidth of a computer network is the maximum amount of data that can be sent in a fixed amount of time.~~  ~~Bandwidth is usually measured in bits per second.~~ | Eks were included above. | | 6.2.2 Explain how the characteristics of the Internet influence the systems built on it. [P4] | 6.2.2A Hierarchy and redundancy help systems scale.  6.2.2B The redundancy of routing (i.e., more than one way to route data) between two points on the Internet increases the reliability of the Internet and helps it scale to more devices and more people.  6.2.2C Hierarchy in the DNS helps that system scale.  6.2.2D Interfaces and protocols enable widespread use of the Internet.  6.2.2E Open standards fuel the growth of the Internet.  6.2.2F The Internet is a packet-switched system through which digital data is sent by breaking the data into blocks of bits called packets, which contain both the data being transmitted and control information for routing the data.  Exclusion Statement (6.2.2F): Specific details of any particular packet switching system are beyond the scope of this course and the AP Exam.  6.2.2G Standards for packets and routing include transmission control protocol/Internet protocol (TCP/IP).  Exclusion Statement (6.2.2G): Specific technical details of how TCP/IP works are beyond the scope of this course and the AP Exam.  6.2.2H Standards for sharing information and communicating between browsers and servers on the Web include HTTP and secure sockets layer/transport layer security (SSL/TLS).  Exclusion Statement (6.2.2H): Understanding the technical aspects of how SSL/TLS works is beyond the scope of this course and the AP Exam.  6.2.2I The size and speed of systems affect their use.  6.2.2J The bandwidth of a system is a measure of bit rate â” the amount of data (measured in bits) that can be sent in a fixed amount of time.  6.2.2K The latency of a system is the time elapsed between the transmission and the receipt of a request. |
| Explain how data is sent through the Internet ~~a computer network~~ via packets. | Information is passed through the internet as a datastream. Datastreams contains chunks of data which are encapsulated in packets.    ~~Data passed through the Internet is broken into sequences of bits called packets.~~  Packets contain ~~both the content of the data and metadata used for routing and reassembling the data.~~ a chunk of data and metadata used for routing a packet between the origin and the destination on the internet and data reassembly  Packets may ~~not arrive or may~~ arrive at the destination in order, ~~or~~ out-of-order, or not at all.  IP, TCP, and UDP are common protocols used on the Internet. | Incorporated TCP/ IP back in at the request of the providers. | | 6.2.2 Explain how the characteristics of the Internet influence the systems built on it. [P4] | 6.2.2A Hierarchy and redundancy help systems scale.  6.2.2B The redundancy of routing (i.e., more than one way to route data) between two points on the Internet increases the reliability of the Internet and helps it scale to more devices and more people.  6.2.2C Hierarchy in the DNS helps that system scale.  6.2.2D Interfaces and protocols enable widespread use of the Internet.  6.2.2E Open standards fuel the growth of the Internet.  6.2.2F The Internet is a packet-switched system through which digital data is sent by breaking the data into blocks of bits called packets, which contain both the data being transmitted and control information for routing the data.  Exclusion Statement (6.2.2F): Specific details of any particular packet switching system are beyond the scope of this course and the AP Exam.  6.2.2G Standards for packets and routing include transmission control protocol/Internet protocol (TCP/IP).  Exclusion Statement (6.2.2G): Specific technical details of how TCP/IP works are beyond the scope of this course and the AP Exam.  6.2.2H Standards for sharing information and communicating between browsers and servers on the Web include HTTP and secure sockets layer/transport layer security (SSL/TLS).  Exclusion Statement (6.2.2H): Understanding the technical aspects of how SSL/TLS works is beyond the scope of this course and the AP Exam.  6.2.2I The size and speed of systems affect their use.  6.2.2J The bandwidth of a system is a measure of bit rate â” the amount of data (measured in bits) that can be sent in a fixed amount of time.  6.2.2K The latency of a system is the time elapsed between the transmission and the receipt of a request. |
| Describe the differences between the Internet and the World Wide Web. | The world wide web is a system of linked pages, programs, and files.  The HTTP protocol is the used on the World Wide Web.  The World Wide Web uses the Internet. | Added HTTP back in based on provider feedback. | | 6.2.1 Explain characteristics of the Internet and the systems built on it. [P5]  6.2.2 Explain how the characteristics of the Internet influence the systems built on it. [P4] | 6.2.1A The Internet and the systems built on it are hierarchical and redundant.  6.2.1B The domain name syntax is hierarchical  6.2.1C IP addresses are hierarchical.  6.2.1D Routing on the Internet is fault tolerant and redundant.  6.2.2A Hierarchy and redundancy help systems scale.  6.2.2B The redundancy of routing (i.e., more than one way to route data) between two points on the Internet increases the reliability of the Internet and helps it scale to more devices and more people.  6.2.2C Hierarchy in the DNS helps that system scale.  6.2.2D Interfaces and protocols enable widespread use of the Internet.  6.2.2E Open standards fuel the growth of the Internet.  6.2.2F The Internet is a packet-switched system through which digital data is sent by breaking the data into blocks of bits called packets, which contain both the data being transmitted and control information for routing the data.  Exclusion Statement (6.2.2F): Specific details of any particular packet switching system are beyond the scope of this course and the AP Exam.  6.2.2G Standards for packets and routing include transmission control protocol/Internet protocol (TCP/IP).  Exclusion Statement (6.2.2G): Specific technical details of how TCP/IP works are beyond the scope of this course and the AP Exam.  6.2.2H Standards for sharing information and communicating between browsers and servers on the Web include HTTP and secure sockets layer/transport layer security (SSL/TLS).  Exclusion Statement (6.2.2H): Understanding the technical aspects of how SSL/TLS works is beyond the scope of this course and the AP Exam.  6.2.2I The size and speed of systems affect their use.  6.2.2J The bandwidth of a system is a measure of bit rate â” the amount of data (measured in bits) that can be sent in a fixed amount of time.  6.2.2K The latency of a system is the time elapsed between the transmission and the receipt of a request. |
| For fault-tolerant systems, like the Internet:  (a) Describe the benefits of fault-tolerance  (b) Explain how a given system is fault-tolerant  (c) Identify vulnerabilities in a system. | * The internet has been engineered to be fault-tolerant, with abstractions for routing and transmitting data. * Redundancy is the inclusion of extra components that can be used to mitigate failure of the system if other components fail. * One way redundancy is accomplished in networks is by having more than one path between two devices. * If a particular device or connection on the internet fails, subsequent data will be sent via a different route, if possible. * When a system can support failures and still continue to function, it is called “fault-tolerant”; this is important because elements of complex systems fail at unexpected times, often in bunches, and fault-tolerance allows users to continue to use the network. * Redundancy within a system often requires additional resources but can provide the benefit of fault-tolerance. * The redundancy of routing between two points on the Internet increases the reliability of the Internet and helps it scale to more devices and more people. | | **NEW — no 2018 comparison**  The Internet has been engineered to be fault-tolerant, with abstractions for routing and transmitting data.   * Routing is the process of finding a path from sender to receiver. * Routing on the Internet is dynamic; it is not specified in advance. * Redundancy is the inclusion of extra components that can be used to mitigate failure of the system if other components fail. * One way redundancy is accomplished in networks is by having more than one path between two devices. * If a particular device or connection on the Internet fails, subsequent data will be sent via a different route, if possible. * When a system can support failures and still continue to function, it is called “fault-tolerant”; this is important because elements of complex systems fail at unexpected times, often in bunches, and fault-tolerance allows users to continue to use the network. * Redundancy within a system often requires additional resources but can provide the benefit of fault-tolerance. * The redundancy of routing between two points on the Internet increases the reliability of the Internet and helps it scale to more devices and more people. | | |
| Compare problem solutions that use sequential, parallel, and distributed computing. | * Sequential computing is a computational model in which operations are performed in order one at a time. * Parallel computing is a computational model where the program is broken into multiple smaller sequential computing operations some of which are performed simultaneously. * Distributed computing is a computational model in which multiple devices are used to run a program. | | **NEW — no 2018 comparison**   * Sequential computing is a computational model in which operations are performed in order one at a time. * Parallel computing is a computational model where the program is broken into multiple smaller sequential computing operations some of which are performed simultaneously. * Distributed computing is a computational model in which multiple devices are used to run a program. | | |
| Determine the efficiency of sequential and parallel solutions. | * Comparing efficiency of solutions can be done by comparing the time it takes them to perform the same task * A sequential solution takes as long as the sum of all of its steps. * A parallel computing solution takes as long as its sequential tasks plus the longest of the tasks done in parallel. * The “speedup” of a parallel solution is measured in the time it took to complete the task sequentially divided by the time it took to complete the task when done in parallel. | | **NEW — no 2018 comparison**   * Performance of sequential computing is typically measured in the time it takes to perform the task. * A parallel computing solution takes as long as its sequential tasks plus the longest of the tasks done in parallel. * Parallel performance is measured in the time it took to complete the task sequentially divided by the time it took to complete the task when done in parallel. | | |
| Describe benefits and challenges of parallel and distributed computing. | * Parallel computing consists of a parallel portion and a sequential portion. * Solutions that use parallel computing can scale more effectively than solutions that use sequential computing. * Distributed computing allows problems to be solved that could not be solved on a single computer, either because of the processing time or storage needs involved. * Distributed computing allows much larger problems to be solved quicker than they could be solved using a single computer. * When increasing the use of parallel computing in a solution, the efficiency of the solution is still limited by the sequential portion. This means at some point, adding parallel portions will no longer increase efficiency. | | **NEW — no 2018 comparison**   * Parallel computing consists of a parallel portion and a sequential portion. * Solutions that use parallel computing can scale more effectively than solutions that use sequential computing. * Distributed computing allows problems to be solved that could not be solved on a single computer, either because of the processing time or storage needs involved. * Distributed computing allows much larger problems to be solved quicker than they could be solved using a single computer. * When increasing the use of parallel computing in a solution, the efficiency of the solution is still limited by the sequential portion. This means at some point, adding parallel portions will no longer increase efficiency. | | |

Big Idea 5 – Impact of Computing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2020 | | Rationale for Changes | 2018 | |
| Learning Objectives | Essential Knowledge |  | Learning Objectives | Essential Knowledge |
| Students must be able to… | Students will know that… |  | Students must be able to… | Students will know that… |
|  | REMOVED |  | 7.5.1 Access, manage, and attribute information using effective strategies. [P1] | 7.5.1A Online databases and libraries catalog and house secondary and some primary resources  7.5.1B Advanced search tools, Boolean logic, and key words can refine the search focus and/or limit search results based on a variety of factors (e.g., peer-review status, type of publication  7.5.1C Plagiarism is a serious offense that occurs when a person presents another's ideas or words as his or her own. Plagiarism may be avoided by accurately acknowledging sources. |
|  | REMOVED |  | 7.5.2 Evaluate online and print sources for appropriateness and credibility [P5] | 7.5.2A Determining the credibility of a soruce requires considering and evaluating the reputation and credentials of the author(s), publisher(s), site owner(s), and/or sponsor(s).  7.5.2B Information from a source is considered relevant when it supports an appropriate claim or the purpose of the investigation |
| Explain how an effect of a computing innovation can be both beneficial and harmful. | ~~Computer scientists~~ People are creators of computing innovations.  As computing evolves, the way ~~we~~ people complete tasks often changes to incorporate new computing innovations.  The total effects of a computing innovation are not always anticipated in advance.  A single effect can be viewed as both positive and negative based on an individual’s perspectives.  Advances in computing as an enabling technology have generated and increased creativity in other fields, such as medicine, engineering, communications and the arts. |  | 7.3.1 Analyze the beneficial and harmful effects of computing. [P4] | 7.3.1A Innovations enabled by computing raise legal and ethical concerns.  7.3.1B Commercial access to music and movie downloads and streaming raises legal and ethical concerns.  7.3.1C Access to digital content via peer to peer networks raises legal and ethical concerns.  7.3.1D Both authenticated and anonymous access to digital information raise legal and ethical concerns.  7.3.1E Commercial and governmental censorship of digital information raise legal and ethical concerns.  7.3.1F Open source and licensing of software and content raise legal and ethical concerns.  7.3.1G Privacy and security concerns arise in the development and use of computational systems and artifacts.  7.3.1H Aggregation of information, such as geolocation, cookies, and browsing history, raises privacy and security concerns.  7.3.1I Anonymity in online interactions can be enabled through the use of online anonymity software and proxy servers.  7.3.1J Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions.  7.3.1K People can have instant access to vast amounts of information online; accessing this information can enable the collection of both individual and aggregate data that can be used and collected.  7.3.1L Commercial and governmental curation of information may be exploited if privacy and other protections are ignored.  7.3.1M Targeted advertising is used to help individuals, but it can be misused at both individual and aggregate levels.  7.3.1N Widespread access to digitized information raises questions about intellectual property.  7.3.1O Creation of digital audio, video, and textual content by combining existing content has been impacted by copyright concerns.  7.3.1P The Digital Millennium Copyright Act (DMCA) has been a benefit and a challenge in making copyrighted digital material widely available.  7.3.1Q Open source and free software have practical, business, and ethical impacts on widespread access to programs, libraries, and code. |
| Describe issues that contribute to the digital divide. | * ~~Online~~ Internet access varies between ~~countries and between~~ socioeconomic, geographic, demographic characteristics and between countries. ~~groups.~~ * The digital divide refers to differing access to computing and the Internet based on socioeconomic, geographic, or demographic characteristics. * The digital divide can affect both groups and individuals. * The digital divide raises issues of equity, access, and influence, both globally and locally. * The digital divide is affected by individuals, organizations and government actions. * ~~The global, socioeconomic, and demographic distribution of computing resources raises issues of equity, access, and influence.~~ * ~~Access to computing and the Internet are impacted by both commercial and governmental initiatives.~~ | order has changed, this is moved later. | 7.4.1 Explain the connections between computing and real-world contexts, including economic, social, and cultural contexts. [P1] | 7.4.1A The innovation and impact of social media and online access is different in different countries and in different socioeconomic groups.  7.4.1B Mobile, wireless, and networked computing have an impact on innovation throughout the world.  7.4.1C The global distribution of computing resources raises issues of equity, access, and power.  7.4.1D Groups and individuals are affected by the 'digital divide' -- differing access to computing and the Internet based on socioeconomic or geographic characteristics.  7.4.1E Networks and infrastructure are supported by both commercial and governmental initiatives. |
| Explain how bias exists in computing innovations. | * Computing innovations can reflect existing human biases because of biases written into the algorithms or biases in the data used to develop the programs. * Programmers should take action to reduce bias in algorithms used for computing innovations as a way of combating existing human biases. * ~~People evaluate or write programs to evaluate the correctness of a program, providing the opportunity to introduce bias.~~ * Biases can be embedded at all levels of software development. |  | **NEW — no 2018 comparison** | |
| Explain how a computing innovation can have an impact beyond its intended purpose. | Computing innovations can be used in ways that the creator had not originally intended. Some examples include:   * The World Wide Web was originally intended only for rapid and easy exchange of information within the scientific community; * Targeted advertising is used to help individuals, but it can be misused at both individual and aggregate levels; * Machine learning and data mining have enabled innovation in medicine, business, and science, but this information could be misused to discriminate against groups of individuals; ~~and~~ * ~~Computing enables innovation by providing the ability to access and share information, but this information, if shared via a cloud network, is only available when network access exists and may be less secure than other methods of sharing information.~~ * Some of the unintended ways computing innovations can be used have a ~~negative~~ harmful impact on society, economy, or culture. * Responsible programmers try to consider the unintended ways their computing innovations can be used and the potential beneficial and harmful effects of these new uses. * It is not~~rarely~~ possible for a programmer to consider all the ~~alternate~~ ways a computing innovation can be used. * Often computing innovations ~~and advances in computing~~ have had a beneficial effect by leading to advances in other fields. * Rapid sharing of the program or the results of running a program with a large number of users can result in significant impacts beyond intended purposes or control of the programmer. |  | 7.1.1 Explain how computing innovations affect communication, interaction, and cognition. [P4] | 7.1.1A Email, short message service (SMS), and chat have fostered new ways to communicate and collaborate.  7.1.1B Video conferencing and video chat have fostered new ways to communicate and collaborate.  7.1.1C Social media continues to evolve and foster new ways to communicate.  Exclusion Statement (7.1.1C): Detailed knowledge of any social media site is beyond the scope of this course and the AP Exam.  7.1.1D Cloud computing fosters new ways to communicate and collaborate.  7.1.1E Widespread access to information facilitates the identification of problems, development of solutions, and dissemination of results.  7.1.1F Public data provides widespread access and enables solutions to identified problems.  7.1.1G Search trends are predictors.  7.1.1H Social media, such as blogs and Twitter, have enhanced dissemination.  7.1.1I Global Positioning System (GPS) and related technologies have changed how humans travel, navigate, and find information related to geolocation.  7.1.1J Sensor networks facilitate new ways of interacting with the environment and with physical systems.  7.1.1K Smart grids, smart buildings, and smart transportation are changing and facilitating human capabilities.  7.1.1L Computing contributes to many assistive technologies that enhance human capabilities.  7.1.1M The Internet and the Web have enhanced methods of and opportunities for communication and collaboration.  7.1.1N The Internet and the Web have changed many areas, including ecommerce, health care, access to information and entertainment, and online learning.  7.1.1O The Internet and the Web have impacted productivity, positively and negatively, in many areas. |
| Explain how people participate in problem-solving processes at scale. | * Widespread access to information and public data facilitates the identification of problems, development of solutions, and dissemination of results. * Science has been impacted by using scale and “citizen science” to solve scientific problems. * Citizen science is scientific research conducted in whole or part by individuals that contribute relevant data to research using their own computing devices. * Crowdsourcing is the practice of obtaining input or information from a large number of people via the Internet. * Human capabilities can be enhanced by collaboration via computing. * Crowdsourcing offers new models for collaboration, such as connecting people with jobs and businesses with funding. |  | 7.1.2 Explain how people participate in a problem solving process that scales. [P4] | 7.1.2A Distributed solutions must scale to solve some problems.  7.1.2B Science has been impacted by using scale and 'citizen science' to solve scientific problems using home computers in scientific research.  7.1.2C Human computation harnesses contributions from many humans to solve problems related to digital data and the Web.  7.1.2D Human capabilities are enhanced by digitally enabled collaboration.  7.1.2E Some online services use the contributions of many people to benefit both individuals and society.  7.1.2F Crowdsourcing offers new models for collaboration, such as connecting people with jobs and businesses with funding.  7.1.2G The move from desktop computers to a proliferation of alwayson mobile computers is leading to new applications. |
| Explain how the use of computing raises legal and ethical concerns. | * Material created on a computer is the intellectual property of the creator or an organization. * Ease of access and distribution of digitized information raises intellectual property concerns regarding ownership, value, and use. * Measures should be taken to safeguard intellectual property. * The use of material created by someone else without permission is plagiarism and may have legal consequences.   Some examples of legal ways to use materials created by someone else include:   * + Creative Commons - a public copyright license that enables the free distribution of an otherwise copyrighted work. This is used when the content creator wants to give others the right to share, use, and build upon the work they have created.   + open source – programs that are made freely available and may be redistributed and modified.   + open access - online research outputs that are free of any and all restrictions on access, and free of many restrictions on use such as copyright or license restrictions. * The use of material created by someone other than yourself should always be cited. * Creative commons, open source, and open access have enabled broad access to digital information. * Using computing to harm individuals or groups of people raise legal and ethical concerns. * Computing can play a role in social and political issues which in turn often raise legal and ethical concerns. * The digital divide raises ethical concerns around computing.   Computing innovations raise legal and ethical concerns. Some examples of these include:   * the development of software that allows commercial access to music and movie downloads and streaming * the development of algorithms that include bias; and * the existence of computing devices that are always listening and recording what you are saying. |  | 7.2.1 Explain how computing has impacted innovations in other fields. [P1] | 7.2.1A Machine learning and data mining have enabled innovation in medicine, business, and science.  7.2.1B Scientific computing has enabled innovation in science and business.  7.2.1C Computing enables innovation by providing access to and sharing of information.  7.2.1D Open access and Creative Commons have enabled broad access to digital information.  7.2.1E Open and curated scientific databases have benefited scientific researchers.  7.2.1F Moore's law has encouraged industries that use computers to effectively plan future research and development based on anticipated increases in computing power.  7.2.1G Advances in computing as an enabling technology have generated and increased the creativity in other fields. |
| Describe the risks to privacy from collecting and storing personal data on a computer system. | Personally identifiable information (PII) is information about an individual that identifies, links, relates, is unique to, or describes them. Examples of PII include:   * social security number; * age; * race; * phone number(s); * medical information; * financial information; and * biometric data. * Search engines can record and maintain a history of searches made by users. * Websites can record and maintain a history of individuals who have viewed their pages. * Devices, websites, and networks can collect information about a user’s location. * Technology enables the collection, use, and exploitation of information about, by, and for individuals, groups, and institutions. * ~~PII can be used to stalk or steal the identity of a person, or to aid in the planning of criminal acts.~~ * Search engines can use search history to suggest websites or for target marketing. * Disparate personal data, such as geolocation, cookies, and browsing history, can be aggregated to create knowledge about an individual. * PII and other information placed online can be used to enhance a user’s online experineces. * PII stored online can be used to simplify making online purchases. * Commercial and governmental curation of information may be exploited if privacy and other protections are ignored. * Information placed online can be used in ways that were not intended and in ways that may have a harmful impact. For example, an email message may be forwarded, tweets can be retweeted, social media posts can be viewed by potential employers. * PII can be used to stalk or steal the identiy of a person, or to aid in the planning of criminal acts. * It is difficult to delete information once it has been placed online. * Applications can collect your location and record where you have been, how you got there, and how long you have been at a given location. * Information posted to social media services can be used by others. Combining information posted on social media and other sources can be used to deduce private information about you. |  | 3.3.1 Analyze how data representation, storage, security, and transmission of data involve computational manipulation of information. [P4]  7.1.1 Explain how computing innovations affect communication, interaction, and cognition. [P4] | 3.3.1A Digital data representations involve trade offs related to storage, security, and privacy concerns.  3.3.1B Security concerns engender tradeoffs in storing and transmitting information.  3.3.1F Security and privacy concerns arise with data containing personal information.  7.1.1A Email, short message service (SMS), and chat have fostered new ways to communicate and collaborate.  7.1.1B Video conferencing and video chat have fostered new ways to communicate and collaborate.  7.1.1C Social media continues to evolve and foster new ways to communicate.  Exclusion Statement (7.1.1C): Detailed knowledge of any social media site is beyond the scope of this course and the AP Exam.  7.1.1D Cloud computing fosters new ways to communicate and collaborate.  7.1.1E Widespread access to information facilitates the identification of problems, development of solutions, and dissemination of results.  7.1.1F Public data provides widespread access and enables solutions to identified problems.  7.1.1G Search trends are predictors.  7.1.1H Social media, such as blogs and Twitter, have enhanced dissemination.  7.1.1I Global Positioning System (GPS) and related technologies have changed how humans travel, navigate, and find information related to geolocation.  7.1.1J Sensor networks facilitate new ways of interacting with the environment and with physical systems.  7.1.1K Smart grids, smart buildings, and smart transportation are changing and facilitating human capabilities.  7.1.1L Computing contributes to many assistive technologies that enhance human capabilities.  7.1.1M The Internet and the Web have enhanced methods of and opportunities for communication and collaboration.  7.1.1N The Internet and the Web have changed many areas, including ecommerce, health care, access to information and entertainment, and online learning.  7.1.1O The Internet and the Web have impacted productivity, positively and negatively, in many areas. |
| Explain how computing resources can be protected and can be misused. | Authentication measures protect devices and information from unauthorized access. Examples of authentication measures include:   * Strong passwords; and * Multi-factor authentication. * A strong password is something that is easy for a user to remember but would be difficult for someone else to guess based on knowledge of that user. * Multi-factor authentication is a method of computer access control in which a user is only granted access after successfully presenting several separate pieces of evidence to an authentication mechanism-typically at least two of the following categories: knowledge (something they know); possession (something they have), and inherence (something they are). * Multi-factor authentication requires at least two steps to unlock protected information; each step adds a new layer of security that must be broken to gain unauthorized access. * Cryptographic protocols and techniques can be used to obscure information transmitted over a network. Examples of cryptographic protocols include   + Symmetric encryption – a method of encryption involving one key for encryption and decryption;   + Public key cryptography – a method of encryption that pairs a public key for encryption and a private key for decryption; and   + Certificate authorities (CAs) – a method that issues digital certificates that validate the ownership of encryption keys used in secured communications and are based on a trust model. Exception: the mathematical methods used in cryptographic are beyond the scope of this course. * Computer virus and malware scanning software can help to protect a computing system against infection. * A computer virus is a malicious program that can copy itself and gain access to a computer in an unauthorized way. Computer viruses often attach themselves to legitimate programs, and start running independently on a computer. * Malware is software intended to damage a computing system or to take partial control over its operation. * All real-world systems have errors or design flaws that hackers can exploit to compromise them. Regular software updates help to fix errors that hackers can exploit to compromise a computing system. * ~~A hacker is a person who has a high level of skill in computer technology or programming. Some hackers use their skills to gain unauthorized access to computer systems or data.~~ * Users can control the permissions applications have for collecting user information. Users should review the permissions of applications to protect their privacy. |  | 6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it. [P1] | 6.3.1A The trust model of the Internet involves tradeoffs.  6.3.1B The domain name system (DNS) was not designed to be completely secure.  6.3.1C Implementing cybersecurity has software, hardware, and human components.  6.3.1D Cyber warfare and cyber crime have widespread and potentially devastating effects.  6.3.1E Distributed denial of service attacks (DDoS) compromise a target by flooding it with requests from multiple systems.  6.3.1F Phishing, viruses, and other attacks have human and software components.  6.3.1G Antivirus software and firewalls can help prevent unauthorized access to private data.  6.3.1H Cryptography is essential to many models of cybersecurity.  6.3.1I Cryptography has a mathematical foundation.  Exclusion Statement (6.3.1I): Specific mathematical functions used in cryptography are beyond the scope of this course and the AP Exam.  6.3.1J Open standards help ensure cryptography is secure.  6.3.1K Symmetric encryption is a method of encryption involving one key for encryption and decryption.  Exclusion Statement (6.3.1K): The methods used in encryption are beyond the scope of this course and the AP Exam.  6.3.1L Public key encryption, which is not symmetric, is an encryption method that is widely used because of the enhanced security associated with its use.  Exclusion Statement (6.3.1L): The mathematical methods used in public key encryption are beyond the scope of this course and the AP Exam.  6.3.1M Certificate authorities (CAs) issue digital certificates that validate the ownership of encrypted keys used in secured communication and are based on a trust model.  Exclusion Statement (6.3.1M): The technical details of the process certificate authorities follow are beyond the scope of this course and the AP Exam. |
| Explain how unauthorized access to computing resources is gained. | Phishing is a technique that is used to trick a user into providing personal information. That personal information can then be used to access sensitive online resources, such as bank accounts and emails.  Keylogging is the use of a program to record every keystroke made by a computer user, in order to gain fraudulent access to passwords and other confidential information.  Data sent over public networks can be intercepted, analyzed and modified. One way that this can happen is through a rogue access point.  A rogue access point is a wireless access point that gives unauthorized access to secure networks.  A malicious link can be disguised on a web page or in an email message.  Unsolicited emails, attachments, links, and forms in emails can be used to compromise the security of a computing system. These can come from unknown senders or from known senders whose security has been compromised.  Untrustworthy (often free) downloads from freeware or shareware sites can contain malware. |  | 7.1.1 Explain how computing innovations affect communication, interaction, and cognition. [P4]  6.3.1 Identify existing cybersecurity concerns and potential options to address these issues with the Internet and the systems built on it. [P1] | 7.1.1A Email, short message service (SMS), and chat have fostered new ways to communicate and collaborate.  7.1.1B Video conferencing and video chat have fostered new ways to communicate and collaborate.  7.1.1C Social media continues to evolve and foster new ways to communicate.  Exclusion Statement (7.1.1C): Detailed knowledge of any social media site is beyond the scope of this course and the AP Exam.  7.1.1D Cloud computing fosters new ways to communicate and collaborate.  7.1.1E Widespread access to information facilitates the identification of problems, development of solutions, and dissemination of results.  7.1.1F Public data provides widespread access and enables solutions to identified problems.  7.1.1G Search trends are predictors.  7.1.1H Social media, such as blogs and Twitter, have enhanced dissemination.  7.1.1I Global Positioning System (GPS) and related technologies have changed how humans travel, navigate, and find information related to geolocation.  7.1.1J Sensor networks facilitate new ways of interacting with the environment and with physical systems.  7.1.1K Smart grids, smart buildings, and smart transportation are changing and facilitating human capabilities.  7.1.1L Computing contributes to many assistive technologies that enhance human capabilities.  7.1.1M The Internet and the Web have enhanced methods of and opportunities for communication and collaboration.  7.1.1N The Internet and the Web have changed many areas, including ecommerce, health care, access to information and entertainment, and online learning.  7.1.1O The Internet and the Web have impacted productivity, positively and negatively, in many areas.  6.3.1A The trust model of the Internet involves tradeoffs.  6.3.1B The domain name system (DNS) was not designed to be completely secure.  6.3.1C Implementing cybersecurity has software, hardware, and human components.  6.3.1D Cyber warfare and cyber crime have widespread and potentially devastating effects.  6.3.1E Distributed denial of service attacks (DDoS) compromise a target by flooding it with requests from multiple systems.  6.3.1F Phishing, viruses, and other attacks have human and software components.  6.3.1G Antivirus software and firewalls can help prevent unauthorized access to private data.  6.3.1H Cryptography is essential to many models of cybersecurity.  6.3.1I Cryptography has a mathematical foundation.  Exclusion Statement (6.3.1I): Specific mathematical functions used in cryptography are beyond the scope of this course and the AP Exam.  6.3.1J Open standards help ensure cryptography is secure.  6.3.1K Symmetric encryption is a method of encryption involving one key for encryption and decryption.  Exclusion Statement (6.3.1K): The methods used in encryption are beyond the scope of this course and the AP Exam.  6.3.1L Public key encryption, which is not symmetric, is an encryption method that is widely used because of the enhanced security associated with its use.  Exclusion Statement (6.3.1L): The mathematical methods used in public key encryption are beyond the scope of this course and the AP Exam.  6.3.1M Certificate authorities (CAs) issue digital certificates that validate the ownership of encrypted keys used in secured communication and are based on a trust model.  Exclusion Statement (6.3.1M): The technical details of the process certificate authorities follow are beyond the scope of this course and the AP Exam. |