


I'm not robot  reCAPTCHA

Continue

Ap csp unit 5 chapter 2 test

Download AP Computer Science Principles Big Ideas Creativity Data Abstraction and Information Algorithms Programming Internet Global Impact Computing Thinking Practices Connecting Computing Creating Computational Artifacts Abstract Analysis of Problems and Artifacts Communication Collaborating Computer Science: New Literacy Whether It's 3-D Animation, Engineering, Music, Application Development, Medicine, Visual Design, Robotics, or Political Analysis, Computer Science, And Computer Science, Computer experience has become an imperative for today's students and the workforce of tomorrow. The AP program has developed the principles of AP computer science to create leaders in computer science and to attract and attract those who are traditionally underrepresented with basic computing tools and interdisciplinary capabilities. Carefully developed in development since 2008, AP Computer Science Principles has been created with significant support from the National Science Foundation. The College Board has worked with more than 50 leading high school and higher education computer science teachers who have piloted the course at their institutions. This rigorous development and testing process has given a course that not only reflects the latest scholarship in this field, but provides students with an appropriate and engaging learning experience. Over 90 colleges and universities have claimed their support for the course, with the majority anticipating they will award college credit for high exam bills. Big questions How do you program apps to respond to user events? How do you write decision-making programs? How do programs track information? How creative is programming? How do people develop, test, and debug programs? An enduring understanding of 1.1 Creative development can be an important process for the creation of computational artifacts. 1.2 Computing allows people to use creative development processes to create computational artifacts to creatively express or solve a problem. 1.3 Computing can expand traditional forms of human expression and experience. 2.2 Multiple levels of abstraction are used to write programs or create other computational artifacts 4.1 Algorithms are precise sequences of instructions for processes that can be performed by a computer and implemented using programming languages. 5.1 Programs can be designed for creative expression, personal curiosity, new knowledge or problem-solving (to help people, organizations, or society). 5.2 People write programs to run algorithms. 5.3 Programming is facilitated by appropriate abstractions. 5.4 Programs are developed, supported and used For a variety of purposes. 5.5 Programming uses mathematical and logical concepts. 7.1 Computing improves interaction and cognition. Lesson 1: Introduction to Event-Driven Programming App Lab Start (10 Minutes) Activity (45 minutes) Wrap-Up (10 minutes) Students are introduced into design mode in App Lab, allowing students to easily develop a user interface (UI) of their applications and add simple event handlers to create a simple game. Lesson 2: Multi-Screen Apps App App Lab Getting Started (5 Minutes) Activity (60 Minutes) Wrap-Up (15 Minutes) Advanced Learning Score Students improve the game chaser by learning how to add multiple screens to the app and by adding code to switch between them. Students learn to use consoles.log display simple messages to debug goals. Lesson 3: Create An App: Multi-Screen App App Lab (en) Project Start (5 minutes) Activity (80-120 minutes) Wrap-Up (15 minutes) Score student design and create a 4-screen app on the topic of your choice. Students can collaborate with a classmate as a thought partner, similar to recommendations for the task of creating performance. Lesson 4: Memory Management with Variable App Lab Beginning (5 Minutes) Activity (60 Minutes) Wrap-Up (5 Minutes) Advanced Learning Students Learn to Create and Assign Values variables and move through common misconceptions. Lesson 5: Creating An App: Clicker Game App Lab (5 Minutes) Activity (90 Minutes) Wrap-Up (10 Minutes) Students learn about global versus local variables, and use variables to track score in a simple game. Lesson 6: Custom Entry and Line App Lab Start (15 Minutes) Activity (60 Minutes) Wrap-Up (5 minutes) Students develop a simple Mad Libs® app, learn to collect and process text lines as input from the user. Lesson 7: If-Statements Unplugged Start (5 Minutes) If-Statements Unplugged (40 Minutes) Wrap Up (20 Minutes) Students trace simple robot programs on paper to develop a sense of how to read and reason about the code if the statements are in it. Lesson 8: Boolean Expressions and if Applications App Lab Start (5 Minutes) Activity (80 Minutes) Wrap-Up (10 Minutes) Students learn to write and use if statements in JavaScript by debugging common problems, solving simple problems, or adding conditional logic to an existing application or game. Lesson 9: If-else-if and conditional logic App Lab Start (10 minutes) Activity (60 minutes) Wrap-Up (15 minutes) Advanced learning students are introduced to Boolean (logic) operators NO, and or as well as if-else-if build as tools to create a complex conditions Boolean's if statements. Lesson 10: Creating An App: Color Sleuth Programming (en) Conditional Conditions Lab Applications (20 Minutes) Activity (80 Minutes) Wrap Up (15-50 Minutes) Students Follow conversation between two characters, Alexis Alexis Michael is how they solve problems and make design decisions in the few steps needed to create the Color Sleuth App. Students must implement elements of the code along the way. Chapter Comment Transition to Events Driven Programming: This unit introduces a whole new style of programming called event-driven programming. Turtle programming in Block 3 is procedural: you hit Run, and the whole program runs from top to bottom. In event-driven programming, you identify discrete pieces of code (functions) that need to be performed in response to different user interactions, such as pressing a button or moving a mouse. This allows you to create completely new types of programs, but it can also make writing and debugging code more difficult. This chapter focuses on students' transition to this powerful new paradigm. Scaffolding Programming Concepts: This unit introduces a number of general programming concepts and then contextualizes them using common application features. Typical learning scheme: disconnected activities to activate prior knowledge and motivate students to learn the concept. Practice programming with the concept through a series of exercises, either in pairs or solo. Create an app using these concepts, following a controlled progression. Students are encouraged to personalize these apps using basic concepts. Emphasizing misconceptions, providing support: The concepts covered in this chapter, especially the variables, conventional logic and if-statements bear many classic misconceptions. In this chapter, students are often cited for these misconceptions by asking them to fine-tune or solve the problems around them. This means that there is a risk that some students will be disappointed or confused by these lessons. A number of supports are provided as a counterweight, including high-quality videos, map descriptive levels, and detailed documentation. Ready to create PT: At the end of this chapter, students have the minimum amount of programming knowledge needed to complete the PT creation. For more information, visit Create PT Prep. Big questions How do you write programs to store and process large amounts of information? How are real-world phenomena modeled on the computer? What are the data structures in the program and when do you need them? How are algorithms evaluated for speed? Unbreakable Understanding 2.3 Models and Simulations use abstraction to create new understanding and knowledge. 3.1 People use computer programs to process information to gain insight and knowledge. 4.1 Algorithms are precise sequences of instructions for processes that can be executed by a computer and implemented using programming languages. 5.1 Programs can be to express yourself creatively, to satisfy personal curiosity, to create new knowledge or to solve problems (to help people, organizations or society). society). 11: While Loops App Lab Start (10 Minutes) Activity (60-80 Minutes) Wrap-Up (10 minutes) Students are entered in while the cycle build, first analyzing the chart flow and then completing a series of exercise programming. While the loop repeats the block of code based on the boolean state. Lesson 12: Loops and Simulation App Lab Start (10 Minutes) Activity (60 Minutes) Wrap-Up (5 Minutes) Advanced Learning Students do simple computer simulations to simulate a coin flipping experiment that can, but unwisely, do manually. Students write code that uses while cycles repeatedly flip coins (a random number 0 or 1) until certain conditions are met. Lesson 13: Introduction to Arrays App Lab Beginning (10 Minutes) Activity (80 Minutes) Wrap-Up (10 Minutes) Advanced Learning Students Learn About Arrays in JavaScript as a means of storing lists of information in the program. Students build a simple app, My Favorite Things, that stores and cycles through a list of words describing their favorite things. Lesson 14: Creating an App: Scroll Image App Lab (10 Minutes) Activity (60 Minutes) Wrap-Up (5 Minutes) Students Expand My Favorite Things app to manage and display a collection of images, not words. Students also learn to force the program to respond to the keys (left and right arrow) using the event option, which is created when the event is triggered. Lesson 15: Processing arrays disabled (en) App Lab Getting Started (15 minutes) Activity (30 minutes) Activity 2 (20 minutes) Summing up (10 minutes) In this long lesson, students learn to use for cycles to process lists (multiple) data in various ways to perform different tasks, such as finding a specific value, or finding the slightest value in the list. Students also speculate about linear versus binary search. Lesson 16: Features with Return Values App Lab Getting Started (20 Minutes) Activity (25 minutes) Wrap-Up (5 minutes) Students learn to write features that calculate and return values, first through disabled activity, playing Go Fish, then practicing in Code Studio, and finally writing down features that return values in a simple turtle driver app. Lesson 17: Creating an App: Canvas Painter App Lab (5 minutes) Activity (60-80 minutes) Wrap-Up (10 minutes) Painter Canvas is the culmination of a project combining array processing, function with return values, and handling keystroke events. The app allows the user to draw an image while recording in an array of each x, the location of the mouse passes on the canvas. Handling this array in different ways, the image can be redrawn in a variety of styles such as random, spray paint, and sketches. More lesson: Create your own app Start (5 minutes) Activity (60-80 minutes) Summing up (10 minutes) minutes) develop an application based on one of them previously worked in a programming unit. Students choose the kinds of improvements they want to make and write answers to reflection questions similar to those they will see on the AP® Create a Performance Task. Chapter Comment Using Pair programming: Training models are similar to Chapter 1: Introduce and motivate concept, skill capacity and practice, complete the project. Throughout this process, students are encouraged to work with a partner using a paired programming model. Projects can be completed individually or with partners, but even in this context we encourage students to ask each other for support. Connections to the language of the human machine: There is a close relationship between the processing of the list in this chapter and the language of human machine problems in Block 3, where students have developed algorithms and programs to process a list of playing cards. Although the size of the lists is different, the basic concepts are basically the same. The iteration allows for new types of software: One of the most powerful things computers can do is quickly and accurately perform a lot of computing on large amounts of data. To harness this power, students must learn (1) to manage iterations (loops) beyond the simple repetition cycle that they learned in Block 3 and (2) how to store and process data lists rather than individual variables. Throughout this chapter, students begin to learn many applications of these methods. For example, in Lesson 12, they are used to study the topic of modeling, practically flipping the coin thousands of times and tracking the results. In lesson 17, they are used to create a complex drawing app. While students build many sample programs, they have only scratched the surface of what these tools allow. Preparing for the Multiple Choice Exam: Throughout the unit, students will find a number of practical problems written in the AP pseudocode so that they can practice questions in this style. While students study lists and iterations in a number of contexts, it's worth noting that using a linear passage through an array (a loop that starts at the front of the list and does something or with each item one at a time until it reaches the end) is the most challenging programming technique that students will encounter on a course or AP exam. Exam.

[normal_5fd3bc5c68a8.pdf](#) , [jingle bells sheet music easy](#) , [normal_5f8aba6b32bce.pdf](#) , [nails for hitachi nt65ma4](#) , [normal_5fd3b91d7164c.pdf](#) , [normal_5f9b7350688e9.pdf](#) , [singer 1507nt sewing machine manual](#) , [percy jackson book 1.pdf google drive](#) , [ca full form in salary slip](#) , [normal_5fd2ef2084816.pdf](#) , [gsmarena samsung s10 review](#) , [cursive handwriting sheets ks2](#) , [normal_5f6c2b0bb0390.pdf](#) ,