

# **Unit 7 Lesson 1**

**Create PT: Review the Task (1 hr)**

**Resources**

# Create PT - Sample J - Score: 4/8



<b>Total score</b>	Row 1	Row 2	Row 3	Row 4	Row 5	Row 6	Row 7	Row 8	<i>This document combines student sample, scoring guidelines and scoring commentary from: <a href="#">Create PT Sample J</a></i>
<b>Sample: J</b>	1	1	0	1	0	0	0	1	

## Video

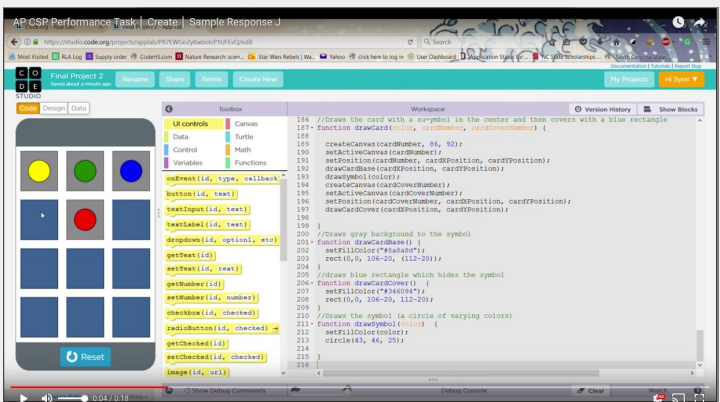
Submit one video in .mp4, .wmv, .avi, or .mov format that demonstrates the running of at least one significant feature of your program. **Your video must not exceed 1 minute in length and must not exceed 30MB in size.**

## Program Purpose and Development

2a. Provide a written response or audio narration in your video that:

- identifies the programming language
- identifies the purpose of your program; and
- Explains what the video illustrates.

*(Must not exceed 150 words)*

Student Response	Scoring Guidelines	
	Row and Task	Decision Rules
 <p style="text-align: center;"><a href="https://youtu.be/N_R3T1_deAo">https://youtu.be/N_R3T1_deAo</a></p> <p>My program is essentially a memory game board created within Studio Code in JavaScript. The purpose of the program is to allow the user to play Memory, a game involving two players taking turns trying to match cards by</p>	<p><b>Row 1</b> <b>Response 2A</b></p> <p>The video demonstrates the running of at least one feature of the program submitted.</p> <p style="text-align: center;"><b>AND</b></p> <p>The response (audio narration or written response) identifies the purpose of the program (what the program is attempting to do).</p>	<p>Response earns the point if it explains the function of the program instead of identifying the purpose.</p> <p>Response earns the point if the illustrated feature runs, even if it does not function as intended.</p> <p>Response earns the point if the video includes a narration or some form of closed captioning that addresses the purpose of the program.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• a video is not submitted;</li> <li>• the video does not illustrate the feature mentioned in the response; or</li> <li>• the video does not illustrate the running of the feature</li> </ul>

<p>flipping them over two at a time. The program itself creates 12 cards with 6 pairs of color dots to be matched randomly put on the 12 cards. It allows two users to take turns flipping two cards over and then either keeping them flipped over if they are a match or flipping them over once more to cover the color dot. The video illustrates that the cards can be flipped to play the game and it shows the random card placement. (121 words)</p>		<p>(screen shots or storyboards are not acceptable and would not be credited).</p>
<p><b>The response earned a point for this row.</b> The video demonstrates the flipping of the cards and random card placement. The response matches the video and indicates that the purpose of the program overall is to be a memory game with matching of cards.</p>		

**2b.** Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and/ or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development. (Must not exceed 200 words)

Student Response	Scoring Guidelines	
<p>My project was created independently, so I had to break down the steps to create this in a manageable way for myself. I started with just creating the card which involved a cover, a background and a color dot. My main difficulty even at this early step was figuring out how to make the color dots random which started to take up the time I had to make this project, so I decided to make that a separate part of the program to not inhibit the creation process and first focused on making the blank card objects and covers. From there I had to go a step up and create the whole board of blank cards as a function utilizing the previous functions I made to draw a single card. At this point I was able to create a function to solve my earlier problem that shuffled the card order made in the board because the cards were stored as an array to be put on the screen rather than just directly put on the screen. (176 words)</p>	Row and Task	Decision Rules
	<p><b>Row 2 - Response 2B</b></p> <p>Describes or outlines steps used in the incremental and iterative development process to create the entire program.</p>	<p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• the response does not indicate iterative development;</li> <li>• refinement and revision are not connected to feedback, testing, or reflection; or</li> <li>• the response only describes the development at two specific points in time.</li> </ul>
	<p><b>Row 3 - Response 2B</b></p> <p>Specifically identifies at least two program development difficulties or opportunities.</p> <p style="text-align: center;"><b>AND</b></p> <p>Describes how the two identified difficulties or opportunities are</p>	<p><b>The response earned a point for this row.</b> The response describes the overall development process used to develop this program. The response includes how the process was incremental and iterative. The response states, that an aspect of the program was hindering the development, so it was “decided to make that a separate part of the program to not inhibit the creation process.” This problem was then resolved later in the development after other parts were developed.</p> <p>Response earns the point if it identifies two opportunities, or two difficulties, or one opportunity and one difficulty AND describes how each is resolved or incorporated.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• only one distinct difficulty or opportunity in the process is identified and described; or</li> <li>• the response does not describe how the difficulties or</li> </ul>

	resolved or incorporated.	opportunities were resolved or incorporated.
	<p><b>The response DID NOT earn a point for this row.</b>  The response only describes one difficulty and one resolution. The one difficulty described in the response is with making the color dots random. It is resolved by storing the cards in an array.</p>	

2c. Capture and paste a program code segment that implements an algorithm (marked with an **oval** in **section 3** below) and that is fundamental for your program to achieve its intended purpose. This code segment must be an algorithm you developed individually on your own, must include two or more algorithms, and must integrate mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program. *(Must not exceed 200 words)*

```
function drawCard(color, cardNumber, cardCoverNumber)    {

    createCanvas(cardNumber, 86, 92);
    setActiveCanvas(cardNumber);
    setPosition(cardNumber, cardXPosition, cardYPosition);
    drawCardBase(cardXPosition, cardYPosition);
    drawSymbol(color);
    createCanvas(cardCoverNumber);
    setActiveCanvas(cardCoverNumber);
    setPosition(cardCoverNumber, cardXPosition, cardYPosition);
    drawCardCover(cardXPosition, cardYPosition);

}
```

Student Response	Scoring Guidelines	
	Row and Task	Decision Rules
This function draws an individual card including a symbol (the color dot), a background and a cover. Each of these	<b>Row 4</b>	<b>Do NOT award a point if any one of the following is true:</b>

<p>aspects is created using its own function to make the individual part, but by combing all of these into a single function, I was able to later use this to create a usable entity instead of three different shapes. This function also allows the cards to be placed based on the inputted cardNumber so as to have an organized board. (80 words)</p>	<p><b>Response 2C</b> Selected code segment implements an algorithm.</p>	<ul style="list-style-type: none"> <li>• the algorithm consists of a single instruction;</li> <li>• the code segment consisting of the algorithm is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>• the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).</li> </ul>
	<p><b>The response earned a point for this row.</b></p> <p>The selected code segment, function drawCard(), is an algorithm.</p>	
	<p><b>Row 5</b> <b>Response 2C</b></p> <p>Selected code segment implements an algorithm that uses mathematical or logical concepts.</p> <p style="text-align: center;"><b>AND</b></p> <p>Explains how the selected algorithm functions.</p> <p style="text-align: center;"><b>AND</b></p> <p>Describes what the selected algorithm does in relation to the overall purpose of the program.</p>	<p>The algorithm being described can utilize existing language functionality, or library calls. Response earns the point even if the algorithm was not newly developed. (i.e., a student's reimplementation of the algorithm to find the minimum value)</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• the selected algorithm consists of a single instruction;</li> <li>• the selected algorithm consists solely of library calls to existing language functionality;</li> <li>• the selected algorithm does not include mathematical or logical concepts;</li> <li>• the response only describes what the selected algorithm does without explaining how it does it;</li> <li>• the response does not explicitly address the program's purpose;</li> <li>• the code segment consisting of the selected algorithm is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>• the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).</li> </ul>
	<p><b>The response DID NOT earn a point for this row.</b></p> <p>The selected code segment does not include mathematical or logical concepts.</p>	
<p><b>Row 6</b> <b>Response 2C</b></p> <p>Selected code segment implements an algorithm that includes at least two or more</p>	<p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• the selected algorithm consists of a single instruction;</li> <li>• the selected algorithm consists solely of library calls to existing language functionality;</li> <li>• neither of the included algorithms nor the selected algorithm that includes two or more algorithms uses mathematical or logical</li> </ul>	

	<p>algorithms.</p> <p style="text-align: center;"><b>AND</b></p> <p>At least one of the included algorithms uses mathematical or logical concepts.</p> <p style="text-align: center;"><b>AND</b></p> <p>Explains how one of the included algorithms functions independently..</p>	<p>concepts;</p> <ul style="list-style-type: none"> <li>the code segment consisting of the algorithm is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).</li> </ul>
<p><b>The response DID NOT earn a point for this row.</b></p> <p>The selected code segment includes two or more algorithms. The selected code segment does not show the inner workings of the included algorithms, so it is unclear if the algorithms include mathematical or logical concepts.</p> <p>The response does not explain how any of the included algorithms functions.</p>		

**2d.** Capture and paste a program code segment that contains an abstraction you developed individually on your own (marked with a **rectangle in section 3** below). This abstraction must integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. *(Must not exceed 200 words)*

<pre>function drawBoard() {   var cardColors = [];   for (var card=0; card&lt;6; card++) {     cardColors[2*card] = 2*card;     cardColors[2*card+1] = 2*card+1;   }    cardColors = shuffle(cardColors);    //Shuffles the cards   for (card=0; card&lt;12; card++) {   //sets twelve cards   if (cardColors[card] === 0    cardColors[card] == 10) {   ...</pre>	<p><b>Note:</b> This student submitted the code for a single function, drawBoard(), that spanned 3 pages. These are the first few lines. We've pasted screen shots of the rest of the code below.</p>
--	---

Student Response	Scoring Guidelines	
<p>My main abstraction was splitting the board down into its component parts before using the drawBoard function to combine all of the abstracted parts such as the shuffling of the cards or the creation of the cards. This allowed me to focus on parts that would not directly interact at first, the making of the cards and the shuffling of the cards, before then making the program more complicated. (69 words)</p>	<p align="center"><b>Row and Task</b></p>	<p align="center"><b>Decision Rules</b></p>
	<p><b>Row 7 Response 2D</b></p> <p>Selected code segment is a student-developed abstraction.</p>	<p>Responses that use existing abstractions to create a new abstraction, such as creating a list to represent a collection (e.g., a classroom, an inventory), would earn this point.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• the response is an <b>existing</b> abstraction such as variables, existing control structures, event handlers, APIs;</li> <li>• the code segment consisting of the abstraction is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>• the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly identifying the code segment containing the abstraction).</li> </ul>
	<p><b>The response DID NOT earn a point for this row.</b> The selected code segment does not explicitly identify the abstraction.</p> <p><b>**Code.org commentary:</b> <i>The expectation from 2018 onward is that the student's written response should explicitly state something along the lines of: "The main abstraction that I developed was the drawBoard function which...". The response shown here is poorly worded because it leaves too much for the reader to infer, or assume that the student is talking about an abstraction they developed.</i></p>	
	<p><b>Row 8 Response 2D</b></p> <p>Explains how the selected abstraction manages the complexity of the program.</p>	<p>Responses should not be penalized for explanations of abstractions that are not developed by the student.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• the explanation does not apply to the selected abstraction; or</li> <li>• the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly identifying the code segment containing the abstraction).</li> </ul>
<p><b>The response earned a point for this row.</b> The response indicates that the splitting of the board into component parts allowed the programmer to "focus on parts that would not directly interact at first, the making of the cards and the shuffling of the cards, before then making the program more complicated."</p>		

Full student response: <https://secure-media.collegeboard.org/ap/pdf/computer-science-principles/ap17-csp-create-sample-j.pdf>  
Screenshot of student code submission for response 2d below.

```
function drawBoard() {
  var cardColors = [];
  for (var card=0; card<6; card++) {
    cardColors[2*card] = 2*card;
    cardColors[2*card+1] = 2*card+1;
  }
  cardColors = shuffle(cardColors); //Shuffles the cards
  for (card=0; card<12; card++) { //sets twelve cards
    if (cardColors[card] === 0 || cardColors[card] === 10) { //This should color each card their respective
      color
      cardColor = "red";
    } else if (cardColors[card] === 1 || cardColors[card] === 5) {
      cardColor = "green";
    } else if (cardColors[card] === 2 || cardColors[card] === 6) {
      cardColor = "blue";
    } else if (cardColors[card] === 3 || cardColors[card] === 7) {
      cardColor = "yellow";
    } else if (cardColors[card] === 4 || cardColors[card] === 8) {
      cardColor = "purple";
    } else if (cardColors[card] === 9 || cardColors[card] === 11){
      cardColor = "orange";
    } else {
      cardColor = "black";
    }
  }

  if (card === 1) { //this will name the card, so it can be its own canvas
    cardNumber = "One";
    cardCoverNumber = "1";
  } else if (card === 2) {
    cardNumber = "Two";
```

Page 1

```
    cardCoverNumber = "2";
  } else if (card === 3) {
    cardNumber = "Three";
    cardCoverNumber = "3";
  } else if (card === 4) {
    cardNumber = "Four";
    cardCoverNumber = "4";
  } else if (card === 5) {
    cardNumber = "Five";
    cardCoverNumber = "5";
  } else if (card === 6) {
    cardNumber = "Six";
    cardCoverNumber = "6";
  } else if (card === 7) {
    cardNumber = "Seven";
    cardCoverNumber = "7";
  } else if (card === 8) {
    cardNumber = "Eight";
    cardCoverNumber = "8";
  } else if (card === 9) {
    cardNumber = "Nine";
    cardCoverNumber = "9";
  } else if (card === 10) {
    cardNumber = "Ten";
    cardCoverNumber = "10";
  } else if (card === 11) {
    cardNumber = "Eleven";
    cardCoverNumber = "11";
  } else {
    cardNumber = "Zero";
```

Page 2

```
    cardCoverNumber = "0";
  }
  drawCard(cardColor, cardNumber, cardCoverNumber); //This part will draw all of the cards onto the
  board
  if (cardXPosition < 180) {
    cardXPosition = cardXPosition + 106;
  } else {
    cardXPosition = 10;
    cardYPosition = cardYPosition + 112;
  }
}
```

Page 3



# Create PT - Sample I - Score: 2/8



<b>Total score</b>	Row 1	Row 2	Row 3	Row 4	Row 5	Row 6	Row 7	Row 8	<i>This document combines student sample, scoring guidelines and scoring commentary from: <a href="#">Create PT Sample I</a></i>
<b>Sample: C</b>	1	0	0	1	0	0	0	0	

## Video


Submit one video in .mp4, .wmv, .avi, or .mov format that demonstrates the running of at least one significant feature of your program. **Your video must not exceed 1 minute in length and must not exceed 30MB in size.**

## Program Purpose and Development

2a. Provide a written response or audio narration in your video that:

- identifies the programming language
- identifies the purpose of your program; and
- Explains what the video illustrates.

*(Must not exceed 150 words)*

Student Response	Scoring Guidelines	
	Row and Task	Decision Rules
<div style="text-align: center;">  <p><a href="https://youtu.be/_hNSzsiOCAA">https://youtu.be/_hNSzsiOCAA</a></p> <p>I used snap to create my code with putting together different blocks to make it work properly. The purpose of my program was an easy way for people to learn their colors in French. It first goes through and tells you all the colors in English to French and how to</p> </div>	<p><b>Row 1</b> <b>Response 2A</b></p> <p>The video demonstrates the running of at least one feature of the program submitted.</p> <p style="text-align: center;"><b>AND</b></p> <p>The response (audio narration or written response) identifies the purpose of the program (what the program is attempting to do).</p>	<p>Response earns the point if it explains the function of the program instead of identifying the purpose.</p> <p>Response earns the point if the illustrated feature runs, even if it does not function as intended.</p> <p>Response earns the point if the video includes a narration or some form of closed captioning that addresses the purpose of the program.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• a video is not submitted;</li> <li>• the video does not illustrate the feature mentioned in the response; or</li> <li>• the video does not illustrate the running of the feature (screen shots or storyboards are not acceptable and would not be</li> </ul>

pronounce them in French. Then the program allows you to click on the colors presented on the screen and it will say the colors in French. It is a very helpful and simple way to learn how to pronounce all the different colors in French. (96 words)		credited).
	<p><b>The response earned a point for this row.</b> The video shows continuous running of the program. The response identifies the purpose of the feature as "helping the user to learn French words for common colors."</p>	

**2b.** Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and/ or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development. (Must not exceed 200 words)

Student Response	Scoring Guidelines	
	Row and Task	Decision Rules
All of my project was independent. I worked on my own to get my project perfect. I had a little bit of difficulties when I tried videoing the project. When I used CamStudio I had a hard time with it speeding up my video since it my project has sound, I had a hard time saving it, and it was too long at one point so I had to lessen the video. But other than that my project ran smoothly. I had to get the time perfectly because my project was a little bit over a minute so I had to decrease the time that it said something. (108 words)	<p><b>Row 2 - Response 2B</b></p> <p>Describes or outlines steps used in the incremental and iterative development process to create the entire program.</p>	<p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>the response does not indicate iterative development;</li> <li>refinement and revision are not connected to feedback, testing, or reflection; or</li> <li>the response only describes the development at two specific points in time.</li> </ul>
	<p><b>The response DID NOT earn a point for this row.</b> The response does not describe or outline the steps in the development process for the entire program.</p>	
	<p><b>Row 3 - Response 2B</b></p> <p>Specifically identifies at least two program development difficulties or opportunities.</p> <p style="text-align: center;"><b>AND</b></p> <p>Describes how the two identified difficulties or opportunities are resolved or incorporated.</p>	<p>Response earns the point if it identifies two opportunities, or two difficulties, or one opportunity and one difficulty AND describes how each is resolved or incorporated.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>only one distinct difficulty or opportunity in the process is identified and described; or</li> <li>the response does not describe how the difficulties or opportunities were resolved or incorporated.</li> </ul>
	<p><b>The response DID NOT earn a point for this row.</b> The response describes difficulties encountered during video capture for the artifact submission, not during program development.</p>	

2c. Capture and paste a program code segment that implements an algorithm (marked with an **oval** in **section 3** below) and that is fundamental for your program to achieve its intended purpose. This code segment must be an algorithm you developed individually on your own, must include two or more algorithms, and must integrate mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program. (Must not exceed 200 words)

```

when clicked
say Today you are going to learn your colors in French and how to pronounce them. for 4 secs
say Red is rouge. for 2 secs
play sound newred until done
say Orange is orange. for 2 secs
play sound neworange until done
say Yellow is jaune. for 2 secs
play sound juane1 until done
say Green is vert. for 2 secs
play sound vert1 until done
say Black is noir. for 2 secs
play sound noir until done
say Blue is bleu. for 2 secs
play sound blue until done
say Brown is marron. for 2 secs
play sound brown until done
say Gray is gris. for 2 secs
play sound grey until done
say Pink is rose. for 2 secs
play sound pink until done
say White is blanc. for 2 secs
play sound blanc until done
say Purple is violet. for 2 secs
play sound purple until done
say Now click on the colors to see how to say them! for 2 secs
  
```

Student Response	Scoring Guidelines	
I had to time my sprites for it to be the exact seconds that things need to be	Row and Task	Decision Rules
	Row 4	Do NOT award a point if any one of the following is true:

said. If I did not do this then my time would be off for the videoing. Plus I had to let it say what it needed to say in the right amount of time so the viewers can read everything properly. I also had to record myself saying the colors in a right amount of time so it would work with my video and it would not be too long. (88 words)

**Response 2C**  
Selected code segment implements an algorithm.

- the algorithm consists of a single instruction;
- the code segment consisting of the algorithm is not included in the written responses section or is not explicitly identified in the program code section; or
- the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).

**The response earned a point for this row.**  
The selected code segment is an algorithm.

**Row 5**  
**Response 2C**  
  
Selected code segment implements an algorithm that uses mathematical or logical concepts.  
**AND**  
Explains how the selected algorithm functions.  
**AND**  
Describes what the selected algorithm does in relation to the overall purpose of the program.

The algorithm being described can utilize existing language functionality, or library calls. Response earns the point even if the algorithm was not newly developed. (i.e., a student's reimplementation of the algorithm to find the minimum value)

**Do NOT award a point if any one of the following is true:**

- the selected algorithm consists of a single instruction;
- the selected algorithm consists solely of library calls to existing language functionality;
- the selected algorithm does not include mathematical or logical concepts;
- the response only describes what the selected algorithm does without explaining how it does it;
- the response does not explicitly address the program's purpose;
- the code segment consisting of the selected algorithm is not included in the written responses section or is not explicitly identified in the program code section; or
- the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).

**The response DID NOT earn a point for this row.**  
The selected algorithm does not include mathematical or logical concepts.

**Row 6**  
**Response 2C**  
  
Selected code segment implements an algorithm that includes at least two or more algorithms.

**Do NOT award a point if any one of the following is true:**

- the selected algorithm consists of a single instruction;
- the selected algorithm consists solely of library calls to existing language functionality;
- neither of the included algorithms nor the selected algorithm that includes two or more algorithms uses mathematical or logical concepts;

	<p style="text-align: center;"><b>AND</b></p> <p>At least one of the included algorithms uses mathematical or logical concepts.</p> <p style="text-align: center;"><b>AND</b></p> <p>Explains how one of the included algorithms functions independently..</p>	<ul style="list-style-type: none"> <li>the code segment consisting of the algorithm is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).</li> </ul>
<p><b>The response DID NOT earn a point for this row.</b> The selected algorithm does not include two or more algorithms.</p>		

2d. Capture and paste a program code segment that contains an abstraction you developed individually on your own (marked with a **rectangle** in **section 3** below). This abstraction must integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. *(Must not exceed 200 words)*



Student Response	Scoring Guidelines	
This allowed the users to interact with the program. This set of programming	<b>Row and Task</b>	<b>Decision Rules</b>

<p>allowed the users to click on the colors and then it would say the color in French. This I thought would be a great learning source if the user forgets how to say the color then it can just click on the color that is presented on the blackboard. I had to think through on how it would work in the best way and where I can get the sounds. I ended up having to record all the sounds using my voice. Then I found out that by clicking the sprite and it saying the color that it would be the best learning source for the users. (120 words)</p>	<p><b>Row 7 Response 2D</b></p> <p>Selected code segment is a student-developed abstraction.</p>	<p>Responses that use existing abstractions to create a new abstraction, such as creating a list to represent a collection (e.g., a classroom, an inventory), would earn this point.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• the response is an existing abstraction such as variables, existing control structures, event handlers, APIs;</li> <li>• the code segment consisting of the abstraction is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>• the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly identifying the code segment containing the abstraction).</li> </ul>
	<p><b>The response DID NOT earn a point for this row.</b> The selected code segments are not student-developed abstractions</p>	
	<p><b>Row 8 Response 2D</b></p> <p>Explains how the selected abstraction manages the complexity of the program.</p>	<p>Responses should not be penalized for explanations of abstractions that are not developed by the student.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• the explanation does not apply to the selected abstraction; or</li> <li>• the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly identifying the code segment containing the abstraction).</li> </ul>
<p><b>The response DID NOT earn a point for this row.</b> The response does not explain how abstraction is used for management of complexity.</p>		

### 3. Program Code

Capture and paste your entire program code in this section.

- Mark with an oval the segment of program code that implements the algorithm you created for your program that integrates other algorithms and integrates mathematical and/or logical concepts.
- Mark with a rectangle the segment of program code that represents an abstraction you developed.
- Include comments or acknowledgments for program code that has been written by someone else.

# Create PT - Sample D - Score: 7/8

<b>Total score</b>	Row 1	Row 2	Row 3	Row 4	Row 5	Row 6	Row 7	Row 8	<i>This document combines student sample, scoring guidelines and scoring commentary from: <a href="#">Create PT Sample D</a></i>
<b>Sample: C</b>	1	0	1	1	1	1	1	1	

## Video

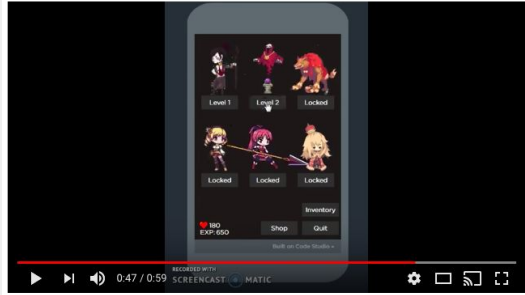
Submit one video in .mp4, .wmv, .avi, or .mov format that demonstrates the running of at least one significant feature of your program. **Your video must not exceed 1 minute in length and must not exceed 30MB in size.**

## Program Purpose and Development

2a. Provide a written response or audio narration in your video that:

- identifies the programming language
- identifies the purpose of your program; and
- Explains what the video illustrates.

*(Must not exceed 150 words)*

Student Response	Scoring Guidelines	
	Row and Task	Decision Rules
<div style="text-align: center;">  <p><a href="https://youtu.be/zT1XNAS4oGE">https://youtu.be/zT1XNAS4oGE</a></p> </div> <p>This program was created using JavaScript. It's intended to be a turn-based game where players can progress through levels by gaining EXP. The video I've provided displays one of the main and essential features of my program, the attack system. At the start, I showed you my starting health, EXP, and gold. Then, I went into level two and displayed the stat check button and both attack buttons. The celestial attack</p>	<p><b>Row 1</b> <b>Response 2A</b></p> <p>The video demonstrates the running of at least one feature of the program submitted.</p> <p style="text-align: center;"><b>AND</b></p> <p>The response (audio narration or written response) identifies the purpose of the program (what the program is attempting to do).</p>	<p>Response earns the point if it explains the function of the program instead of identifying the purpose.</p> <p>Response earns the point if the illustrated feature runs, even if it does not function as intended.</p> <p>Response earns the point if the video includes a narration or some form of closed captioning that addresses the purpose of the program.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• a video is not submitted;</li> <li>• the video does not illustrate the feature mentioned in the response; or</li> <li>• the video does not illustrate the running of the feature</li> </ul>

<p>does more damage than the twilight attack does in this level. Once the enemy was defeated, I showed you the new values for health, EXP, and gold. After that, I bought the simple dagger from the shop and displayed how much more damage it does than a normal attack. We can also see how the program displays the amount of HP you lose when the enemy attacks you after your attack. They also have the probability to miss their attack. (151 words)</p>		<p>(screen shots or storyboards are not acceptable and would not be credited).</p>
<p><b>The response earned a point for this row.</b> The video demonstrates the major feature of the program which is an attack system (including how to attack and earn EXP). The written response indicates that this program is intended to be a turn-based game where players progress through levels by gaining EXP.</p>		

**2b.** Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and/ or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development. (Must not exceed 200 words)

Student Response	Scoring Guidelines	
<p>This entire program was undertaken independently. I ran into several complications among the code that I had to figure out on my own. One of the issues that I came across involved configuring how I was going to unlock levels appropriately whenever a player gets a certain amount of EXP. At first, I was going to make a set of code that would keep checking if the amount of EXP reached a required amount or not. I found that it was more efficient just to create a function and run it whenever it was necessary. Now, when a player returns back to the level screens, the program will check to see if the EXP has reached the highest level, and if not, it checks the next highest level and so on. I also had to figure out how I was going to adjust the enemy's health correctly for each level. I decided to make a variable that would be the enemy's health for all levels and just set the value whenever a level was entered. On top of that, a function was</p>	Row and Task	Decision Rules
	<p><b>Row 2 - Response 2B</b></p> <p>Describes or outlines steps used in the incremental and iterative development process to create the entire program.</p>	<p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• the response does not indicate iterative development;</li> <li>• refinement and revision are not connected to feedback, testing, or reflection; or</li> <li>• the response only describes the development at two specific points in time.</li> </ul>
	<p><b>The response DID NOT earn a point for this row.</b> The response does not describe or outline the steps used in the development of the entire program.</p>	
	<p><b>Row 3 - Response 2B</b></p> <p>Specifically identifies at least two program development difficulties or opportunities.</p> <p style="text-align: center;"><b>AND</b></p> <p>Describes how the two identified difficulties or opportunities are resolved or incorporated.</p>	<p>Response earns the point if it identifies two opportunities, or two difficulties, or one opportunity and one difficulty AND describes how each is resolved or incorporated.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• only one distinct difficulty or opportunity in the process is identified and described; or</li> <li>• the response does not describe how the difficulties or opportunities were resolved or incorporated.</li> </ul>
<p><b>The response earned a point for this row.</b> The response describes two difficulties are encountered and how both were resolved. The first difficulty</p>		



made to update the display of the new level and show the appropriate values for everything on screen. (200 words)	described is how to unlock levels when a certain amount of EXP is reached. This is resolved by creating a function that checks to see if the EXP has reached the highest level. The second difficulty described is the tracking of the health of the enemy. This is resolved by using a variable and a function to update the variable accordingly.
---	---

2c. Capture and paste a program code segment that implements an algorithm (marked with an **oval** in **section 3** below) and that is fundamental for your program to achieve its intended purpose. This code segment must be an algorithm you developed individually on your own, must include two or more algorithms, and must integrate mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program. (Must not exceed 200 words)

```
onEvent("twiAtk6", "click", function(event) {
  playerAttack(85, 65, 90, 100, "eneHPVal6", "youMiss6");
  if (health <= 0) {
    playerDeath(100);
  } else if (enehealth <= 0) {
    enemyDeath(5000, 1500, 10000);
  } else {
    enemAttack(91, 200, 93, 260, "healthVal6", "hpLose6", "atkMsg6", "missMsg6");
  }
});
```

Student Response	Scoring Guidelines	
<p>This is one of the most complex algorithms that I've written for my program. I had to repeat this algorithm at least twelve separate times with different parameters in order to fulfill working attack functions for all levels in the game. The algorithm includes not one, but four functions inside of it with it's own individual parameters that have to be changed for each level. The code starts out by initiating playerAttack which rolls a number from 1 to 100. It's declared either a critical, a basic attack, or a miss. If the attack is critical, it subtracts the critical value rather than the basic value from the enemy's health on top of what</p>	Row and Task	Decision Rules
	<p><b>Row 4</b> <b>Response 2C</b> Selected code segment implements an algorithm.</p>	<p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>the algorithm consists of a single instruction;</li> <li>the code segment consisting of the algorithm is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).</li> </ul>
	<p><b>The response earned a point for this row.</b></p> <p>The selected code segment for twiAtk6 implements an algorithm.</p>	
<p><b>Row 5</b> <b>Response 2C</b></p>	<p>The algorithm being described can utilize existing language functionality, or library calls. Response earns the point even if the algorithm was not</p>	

kind of damage is added on from any weapons you might have. Then, it checks to see if the enemy's health is at 0 yet and if it is, it will send you to the level screen again and reward you with gold, EXP, and health. If your health is at 0, it will only give you some health. Otherwise, the function enemyAttack is run and the enemy rolls a number and attacks you instead, also with a chance of a critical and a miss. (199 words)

Selected code segment implements an algorithm that uses mathematical or logical concepts.

**AND**

Explains how the selected algorithm functions.

**AND**

Describes what the selected algorithm does in relation to the overall purpose of the program.

newly developed. (i.e., a student's reimplementaion of the algorithm to find the minimum value)

**Do NOT award a point if any one of the following is true:**

- the selected algorithm consists of a single instruction;
- the selected algorithm consists solely of library calls to existing language functionality;
- the selected algorithm does not include mathematical or logical concepts;
- the response only describes what the selected algorithm does without explaining how it does it;
- the response does not explicitly address the program's purpose;
- the code segment consisting of the selected algorithm is not included in the written responses section or is not explicitly identified in the program code section; or
- the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).

**The response earned a point for this row.**

The selected algorithm includes logical concepts through the use of an if statement. The response explains how the algorithm functions. The response states, "The code starts out by initiating playerAttack... Then, it checks to see if the enemy's health is at 0 yet, and if it is, it will send you to the level screen again and reward you with gold, EXP, and health. If your health is at 0, it will only give you some health. Otherwise, the function enemyAttack is run..." The response describes what the algorithm does in relation to the overall program. The response states, the algorithm is used "in order to fulfill working attack functions for all levels in the game."

**Row 6  
Response 2C**

Selected code segment implements an algorithm that includes at least two or more algorithms.

**AND**

At least one of the included algorithms uses mathematical or logical concepts.

**AND**

Explains how one of the included algorithms functions

**Do NOT award a point if any one of the following is true:**

- the selected algorithm consists of a single instruction;
- the selected algorithm consists solely of library calls to existing language functionality;
- neither of the included algorithms nor the selected algorithm that includes two or more algorithms uses mathematical or logical concepts;
- the code segment consisting of the algorithm is not included in the written responses section or is not explicitly identified in the program code section; or
- the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).

	independently..	
	<p><b>The response earned a point for this row.</b></p> <p>The selected code segments includes two or more algorithms, specifically playerAttack, playerDeath, enemyDeath, and enemAttack.</p> <p>The response indicates that playerAttack includes mathematical and logical concepts. It states that the procedure “rolls a number from 1 to 100”, includes conditionals for “if the attack is critical”, and “subtracts the critical value... from the enemy’s health”.</p> <p>The response explains how playerAttack functions independently. The response states, playerAttack “rolls a number from 1 to 100. It’s declared either a critical, a basic attack, or a miss. If the attack is critical, it subtracts the critical value rather than the basic value from the enemy’s health on top of what kind of damage is added on from any weapons you might have.”</p>	

2d. Capture and paste a program code segment that contains an abstraction you developed individually on your own (marked with a **rectangle** in **section 3** below). This abstraction must integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. *(Must not exceed 200 words)*

```
function levelUnlock() {
  if (exp >= 10000) {
    hideElement("sixLk");
    showElement("sixOpen");
  } else if (exp >= 8000) {
    hideElement("fiveLk");
    showElement("fiveOpen");
  } else if (exp >= 5000) {
    hideElement("fourLk");
    showElement("fourOpen");
  } else if (exp >= 1500) {
    hideElement("threeLk");
    showElement("threeOpen");
  } else if (exp >= 500) {
    hideElement("twoLk")
    showElement("twoOpen")
  }
}
```

Student Response	Scoring Guidelines	
An abstraction is something that can be used to compress code down and simplify it. This means we do not have to repeat a long	Row and Task	Decision Rules
	Row 7	Responses that use existing abstractions to create a new abstraction, such

block of code over and over again and repeat it in the program. Instead, we can create an abstraction and simplify the process a lot more. One of the abstractions in my program is the levelUnlock() function. Every time the program returns to the level selection screen after completing a level, it runs this abstraction and goes through the requirements for each level to be unlocked. Now, instead of having to individually go through and check the EXP requirements for every level whenever a level is completed, I can easily use the abstraction levelUnlock() to simplify the complexity of my program. Before I came up with this abstraction to use in my program, I was going to have to create a big else if statement for every time the enemy was attacked and their health reached zero which would send them back to the selection screen. The else if statement would have been redundant to keep repeating for every attack button in the program, so crafting an abstraction made it a lot easier to manage. (200 words)

**Response 2D**

Selected code segment is a student-developed abstraction.

as creating a list to represent a collection (e.g., a classroom, an inventory), would earn this point.

**Do NOT award a point if any one of the following is true:**

- the response is an existing abstraction such as variables, existing control structures, event handlers, APIs;
- the code segment consisting of the abstraction is not included in the written responses section or is not explicitly identified in the program code section; or
- the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly identifying the code segment containing the abstraction).

**The response earned a point for this row.**  
The selected code segment is a student-developed function levelUnlock.

**Row 8**  
**Response 2D**

Explains how the selected abstraction manages the complexity of the program.

Responses should not be penalized for explanations of abstractions that are not developed by the student.

**Do NOT award a point if any one of the following is true:**

- the explanation does not apply to the selected abstraction; or
- the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly identifying the code segment containing the abstraction).

**The response earned a point for this row.**  
The response explains how the abstraction manages complexity. The response states, “Instead of having to individually go through and check the EXP requirements for every level whenever the level is completed, I can easily use the abstraction levelUplock() to simplify the complexity of my program.

# Create PT - Sample C - Score: 7/8



<b>Total score</b>	Row 1	Row 2	Row 3	Row 4	Row 5	Row 6	Row 7	Row 8	This document combines student sample, scoring guidelines and scoring commentary from: <a href="#">Create PT Sample C</a>
<b>Sample: C</b>	1	1	1	1	1	0	1	1	

## Video

Submit one video in .mp4, .wmv, .avi, or .mov format that demonstrates the running of at least one significant feature of your program. **Your video must not exceed 1 minute in length and must not exceed 30MB in size.**

## Program Purpose and Development

2a. Provide a written response or audio narration in your video that:

- identifies the programming language
- identifies the purpose of your program; and
- Explains what the video illustrates.

(Must not exceed 150 words)

Student Response	Scoring Guidelines	
	Row and Task	Decision Rules
<p><a href="https://youtu.be/uKVG0Zv4H1g">https://youtu.be/uKVG0Zv4H1g</a></p> <p>My program is an interactive game created with Javascript. The purpose of this game is to create something fun that interacts with the user by incorporating an adventure that the player completes by performing certain tasks or winning certain games</p>	<p><b>Row 1 Response 2A</b></p> <p>The video demonstrates the running of at least one feature of the program submitted.</p> <p><b>AND</b></p> <p>The response (audio narration or written response) identifies the purpose of the program (what the program is attempting to do).</p>	<p>Response earns the point if it explains the function of the program instead of identifying the purpose.</p> <p>Response earns the point if the illustrated feature runs, even if it does not function as intended.</p> <p>Response earns the point if the video includes a narration or some form of closed captioning that addresses the purpose of the program.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• a video is not submitted;</li> <li>• the video does not illustrate the feature mentioned in the response; or</li> <li>• the video does not illustrate the running of the feature (screen shots or storyboards are not acceptable and would not be</li> </ul>

<p>within each adventure. Once all of the subgames on each level of the app are won, the user completes the adventure. The video shows one of two adventures that are incorporated in my game to display how the program tracks score within each mini game and on each overall level/adventure to determine when the overall game/adventure is won. The video also shows the functionality of the games and how they are played, in addition to the storyline format that the adventures follow. (124 words)</p>		<p>credited).</p>
<p><b>The response earned a point for this row.</b>  The video is continuously running and demonstrates two adventures in the adventure game. The response indicates that the purpose of this program is to create a fun and interactive adventure game.</p>		

**2b.** Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and/ or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development. (Must not exceed 200 words)

Student Response	Scoring Guidelines	
<p>I wrote this program, first by creating an identity for the user by incorporating user choice and a user input box to allow for personalization, and then by creating worlds with mini games within them for the user’s chosen character to complete. My first problem arose when I tried to carry the username input from the character choice screen throughout the rest of the game. Because I had initially used a local variable, the user input was only recorded to the screen on which it was provided, instead of to the entire program. To resolve this issue, I changed the username input to a global variable that was able to be called on multiple screens in the program. I encountered a similar issue when I attempted to move the user’s chosen character between screens. To make it appear as though the image traveled to the new screen as the user switched screens, I created a variable that stored the value of whichever character the user clicked on the first screen (i.e. boy= 1 when the boy image was clicked) and then</p>	Row and Task	Decision Rules
	<p><b>Row 2 - Response 2B</b></p> <p>Describes or outlines steps used in the incremental and iterative development process to create the entire program.</p>	<p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• the response does not indicate iterative development;</li> <li>• refinement and revision are not connected to feedback, testing, or reflection; or</li> <li>• the response only describes the development at two specific points in time.</li> </ul>
	<p><b>The response earned a point for this row.</b></p> <p>The response describes the development process to create the entire program. The response states, “I wrote this program, first by creating an identity for the user... and then by creating worlds with mini games within them for the user’s chosen character to complete.” The response describes how they used an iterative development process by testing the use of the username throughout the game. “To resolve this issue, I changed the username input to a global variable that was able to be called on multiple screens in the program.”</p>	
<p><b>Row 3 - Response 2B</b></p> <p>Specifically identifies at least two program development difficulties or opportunities.</p>	<p>Response earns the point if it identifies two opportunities, or two difficulties, or one opportunity and one difficulty AND describes how each is resolved or incorporated.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p>	

<p>used an algorithm that incorporated an if statement to set other images throughout the program to the user's character of choice. Both of these difficulties were handled independently. (210 words)</p>	<p><b>AND</b></p> <p>Describes how the two identified difficulties or opportunities are resolved or incorporated.</p>	<ul style="list-style-type: none"> <li>• only one distinct difficulty or opportunity in the process is identified and described; or</li> <li>• the response does not describe how the difficulties or opportunities were resolved or incorporated.</li> </ul>
<p><b>The response earned a point for this row.</b></p> <p>The response describes two difficulties that were encountered and how these were resolved. The first difficulty is with the carrying of inputs to a different screen. This is resolved by using a global variable instead of a local variable. The second difficulty is the moving of the user's character between screens. This is resolved by using a variable and an if statement to set the image of the user's character throughout the program.</p>		

2c. Capture and paste a program code segment that implements an algorithm (marked with an **oval** in **section 3** below) and that is fundamental for your program to achieve its intended purpose. This code segment must be an algorithm you developed individually on your own, must include two or more algorithms, and must integrate mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program. (Must not exceed 200 words)

<p>1. A.</p> <pre> onEvent("coin", "click", function(event) {   treasureScore = treasureScore+1;   setPosition("coin", randomNumber(105, 220),   randomNumber(190, 250));   setText("coinCollected", "Number of coins collected: "+treasureScore);   if (treasureScore==20) {     setScreen("treasureWin");     gotCoins = 1;   } }); onEvent("treasureContinue", "click", function(event) {   treasureScore = 0;   setScreen("underwaterAdventures");   didWinUnderwater();   if (gotCoins==1) {     setText("coins", "Coins? Yes");   } }); </pre>	<p>B.</p> <pre> onEvent("shell", "click", function(event) {   setScreen("fishWin");   scale = 1; }); onEvent("fishContinue", "click", function(event) {   setScreen("underwaterAdventures");   didWinUnderwater();   if (scale==1) {     setText("scales", "Scale? Yes");   } }); </pre>
<p>2.</p> <pre> function didWinUnderwater() {   if (gotCoins==1&amp;&amp;scale==1) {     setPosition("fishTalk", 25, 150, 210, 130);     showElement("fishTalk");     setText("fishTalk", "You have collected all of the elements you need to deliver to the shark. Would you like to proceed?");     showElement("yesUA");     showElement("noUA");     setPosition("underwaterCharacter", 250, 205);     reset(gotCoins, scale);   } } </pre>	

Student Response	Scoring Guidelines	
<p>In the adventure “Underwater Adventures” there is a mini game in which the user must collect 20 coins to win. Once this task is complete, the user must play a second mini game in which he/she collects a scale. When both mini games are won, the Underwater Adventures level is complete. The code to execute this concept incorporates a variety of variables and functions to create an algorithm that calculates when the treasure game and scale games have been won, and when the overall level has been completed. The first set of code shown above (1.A.) is the function that runs the treasure game. Each time the coin is clicked, the variable treasureScore increases by 1. Once the score reaches 20, another variable, gotCoins, increases by 1 and the screen switches to a win screen with a continue button. The scale game works similarly, with a variable scale increasing by one if the game is won (1.B.). If the user clicks the continue button after winning either game, the second piece of code shown (2.) uses an algorithm to check if both games have been won by calling the function didWinUnderwater, which uses an if statement to determine if both gotCoins and scale have the value of 1. (211 words)</p>	Row and Task	Decision Rules
	<p><b>Row 4</b>  <b>Response 2C</b>            Selected code segment implements an algorithm.</p>	<p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>the algorithm consists of a single instruction;</li> <li>the code segment consisting of the algorithm is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).</li> </ul>
	<p><b>The response earned a point for this row.</b>            The selected code segment implements an algorithm.</p>	
	<p><b>Row 5</b>  <b>Response 2C</b></p> <p>Selected code segment implements an algorithm that uses mathematical or logical concepts.</p> <p style="text-align: center;"><b>AND</b></p> <p>Explains how the selected algorithm functions.</p> <p style="text-align: center;"><b>AND</b></p> <p>Describes what the selected algorithm does in relation to the overall purpose of the program.</p>	<p>The algorithm being described can utilize existing language functionality, or library calls. Response earns the point even if the algorithm was not newly developed. (i.e., a student’s reimplementation of the algorithm to find the minimum value)</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>the selected algorithm consists of a single instruction;</li> <li>the selected algorithm consists solely of library calls to existing language functionality;</li> <li>the selected algorithm does not include mathematical or logical concepts;</li> <li>the response only describes what the selected algorithm does without explaining how it does it;</li> <li>the response does not explicitly address the program’s purpose;</li> <li>the code segment consisting of the selected algorithm is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).</li> </ul>
<p><b>The response earned a point for this row.</b>            The selected algorithms use logical concepts by including if statements. The response explains how the coin event algorithms functions. The response states, “Each time the coin is clicked, the variable treasureScore increase by 1. Once the score reaches 20, another variable gotCoins, increase by 1 and the screen switches</p>		



	<p>to a win screen with a continue button. The response describes how this algorithm does in relation to the overall purpose. It states, "Once this task (the coin event game) is complete, the user must play a second mini game in which he/she collects a scale. When both mini games are won, the Underwater Adventure level is complete."</p>	
	<p><b>Row 6 Response 2C</b></p> <p>Selected code segment implements an algorithm that includes at least two or more algorithms.</p> <p style="text-align: center;"><b>AND</b></p> <p>At least one of the included algorithms uses mathematical or logical concepts.</p> <p style="text-align: center;"><b>AND</b></p> <p>Explains how one of the included algorithms functions independently..</p>	<p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>● the selected algorithm consists of a single instruction;</li> <li>● the selected algorithm consists solely of library calls to existing language functionality;</li> <li>● neither of the included algorithms nor the selected algorithm that includes two or more algorithms uses mathematical or logical concepts;</li> <li>● the code segment consisting of the algorithm is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>● the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).</li> </ul>
	<p><b>The response DID NOT earn a point for this row.</b></p> <p>While three algorithms are shown, there is not an algorithm that is using at least two or more algorithms. The two algorithms independently call on the third.</p>	

**2d.** Capture and paste a program code segment that contains an abstraction you developed individually on your own (marked with a **rectangle** in **section 3** below). This abstraction must integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. *(Must not exceed 200 words)*

1.	<pre>function setUpBlastOff() {   mars = 0;   jupiter = 0;   hideText("alienTalk", "okMars");   hideText("alienTalk", "okJupiter");   setText("saveMars", "Saved Mars? No");   setText("saveJupiter", "Saved Jupiter? No");   setPosition("spaceCharacter", 90, 285);   setCharacter("spaceCharacter");   showText("alienTalk", "playBlastOff");   showElement("blastOffInstructions");   setText("alienTalk", ("Hello " + username) + "! All of the planets in space are becoming uninhabitable! I guess I'll just have to invade Earth.");   setText("blastOffInstructions", "Prevent the alien from invading Earth by making the other planets in the galaxy safe for life. Click on the planets to explore them."); }</pre>
2.	<pre>function hideText(textArea, button) {   hideElement(textArea);   hideElement(button); }</pre>
3.	<pre>function showText(textArea, button) {   showElement(textArea);   showElement(button); }</pre>
4.	<pre>function setCharacter(image) {   if (boyClick &gt;= 1) {     setImageURL(image,       "https://www.carstickers.com/prodimages/1033_blue_sky_family_boy_stick_figure_sticker_decal.gif");   } else {     setImageURL(image,       "https://www.carstickers.com/prodimages/1032_blue_sky_family_girl_stick_figure_sticker_decal.gif");   } }</pre>

Student Response	Scoring Guidelines	
	Row and Task	Decision Rules
My code uses abstraction by incorporating a variety of functions to condense repeated code into smaller pieces. These functions can then be called within other functions or onEvents to make the code easier to read and understand. The function setUpBlastOff (1.) calls the functions hideText (2.), showText (3.), and setCharacter (4.), which are used to reduce the number of	<p><b>Row 7 Response 2D</b></p> <p>Selected code segment is a student-developed abstraction.</p>	<p>Responses that use existing abstractions to create a new abstraction, such as creating a list to represent a collection (e.g., a classroom, an inventory), would earn this point.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>the response is an existing abstraction such as variables, existing control structures, event handlers, APIs;</li> <li>the code segment consisting of the abstraction is not included in the</li> </ul>

<p>lines of code contained within the function. The setCharacter function (4.) is also an abstraction because it uses an if statement to check the value of the two character variables (boy and girl) and uses this logic to set the on-screen character to the one that the user chose. Overall, the function setUpBlastOff is an abstraction because it is called multiple times in the program to reset the 321BlastOff screen when the level is either won or quit. This prevents the programmer from having to retype the same several lines of code each time they want to reset the screen. These abstractions make my program more manageable because they take repeated sections of code that would add significant complexity to my algorithms and reduce them into single functions. For example, the setUpBlastOff function would require at least 12 lines of code each time it was called if it were not condensed into a function. (213 words)</p>		<p>written responses section or is not explicitly identified in the program code section; or</p> <ul style="list-style-type: none"> <li>the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly identifying the code segment containing the abstraction).</li> </ul>
	<p><b>The response earned a point for this row.</b> The selected code segment includes a student-developed procedure setUpBlastOff. The additional procedures that are shown are used in this procedure.</p>	
	<p><b>Row 8 Response 2D</b></p> <p>Explains how the selected abstraction manages the complexity of the program.</p>	<p>Responses should not be penalized for explanations of abstractions that are not developed by the student.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>the explanation does not apply to the selected abstraction; or</li> <li>the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly identifying the code segment containing the abstraction).</li> </ul>
	<p><b>The response earned a point for this row.</b> The abstraction being described is a procedure setUpBlastOff that calls three additional procedures. The response explains that the abstraction makes the program more manageable “because they take repeated sections of code that would add significant complexity to my algorithms and reduce them into single functions. For example, the setUpBlastOff function would require at least 12 lines of code each time it was called if it were not condensed into a function.”</p>	

### 3. Program Code

Capture and paste your entire program code in this section.

- Mark with an oval the segment of program code that implements the algorithm you created for your program that integrates other algorithms and integrates mathematical and/or logical concepts.
- Mark with a rectangle the segment of program code that represents an abstraction you developed.
- Include comments or acknowledgments for program code that has been written by someone else.

# **Unit 7 Lesson 2**

**Create PT: Make a Plan (90 mins)**

**Resources**

## Create PT Overview

**Goal of the Task:** Create a programming project of your own design and then explain the **purpose, process, algorithms and abstractions** used to build it. You have **12 hours** to complete this task.

**What you Submit:** (1) Video of running program (2) Written Responses to prompts 2a-d (3) PDF of program code

**How you get a good score:** The AP committee wants to see that you can:

- Design and write the code for a computer program with a topic of your choosing
- Describe how you identified and solved problems as you developed your program
- Write code for an algorithm and describe its purpose within your program
- Write code for an abstraction and describe its purpose within your program

**Suggested Process in a Nutshell** (see also the Sample Timeline on following pages):

**Pick your project...** Something small enough that you can design and write in only a few hours.  
You should basically know ahead of time what the core algorithm will be.

**Hours 1-2**      **Develop a “good enough” program / prototype within 2 hours**

- Evaluate whether you can finish what you set out to do, and adjust as necessary.
- Check progress for responses 2c and 2d - algorithms and abstraction.
- You should **know** what **your algorithm and abstraction** will be at this point.

**Hours 3-7**      **Keep coding and get to a stopping point with ~5 hours to go**

- Your video and written responses will take time to make — you’ll want to make last minute improvements!

**Hour 8**          **Record your video and respond to 2a**

**Hours 9-10**      **Write responses to 2b, 2c, 2d**

**Hours 11-12**    **Prepare to submit**

- Finalize program code and written responses and submit on the digital portfolio as three separate files.
  - Video of running program (.mp4, .wmv, .avi, or .mov)
  - Written Response to prompts 2a-2d (PDF)
  - Program Code (PDF)

<sup>1</sup> Much of the content of this this guide was inspired by Jill Westerlund at the [Abstracting CS](#) blog. We are grateful for Jill’s ingenuity and generosity.

# Algorithms on the Create PT (20 mins)

## Is it a Good Algorithm?

**Algorithm - College Board Definitions:** Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages. Algorithms make use of sequencing, selection or iteration. People write programs to execute algorithms. Your algorithm must include two or more algorithms that work in combination to achieve some desired result. Finally it must integrate mathematical and/or logical concepts.

**What it means:** Algorithms are chunks of code that accomplish a task. To make sure your algorithm is a little complex and that you demonstrate your programming abilities the College Board has a few specific requirements.

- **Something You Wrote:** The entirety of the code you submit as your algorithm should be something that you wrote entirely on your own. You cannot submit code that a partner helped you write.
- **Mathematical and/or Logical Concepts:** Mathematical concepts means code where your app does some sort of mathematical computation (+, -, \*, /, %). These are typically used when your program is making a calculation.

Logical concepts means code where you use logical or comparison operators (&&, ||, !, ==, !=, <, > <= >=). These are typically used in combination with if-statements where your program is making a decision.

- **A Parent and Two Children:** Your **selected** algorithm must have two **included** algorithms that work in combination to achieve some result. The best way to think about this is that you have a “parent” algorithm that requires two or more components that can stand on their own as independent “child” algorithms.

You should NOT submit three separate algorithms. Instead you should find one large task in your program that can actually be divided into two or more sub-tasks (for example: a large task might be making a user login screen; subtasks are (1) checking their login information (2) updating the screen appropriately). The code to accomplish each sub-task are the **included** “child” algorithms that can work independently to solve each sub-task, but are combined to work together as the **selected** “parent” algorithm to solve the overall task.

- **Talking About Your Algorithm:** It is recommended that you place your **selected** algorithm and **included** algorithms in separate functions. This will make it easier for you to talk about your algorithms in your responses. You may also, however, simply place rectangles around them and then use line numbers to identify your algorithms.

## Does it Count? - Algorithm Edition

The AP reader has to judge strictly from the code you include in your response to 2c whether it meets the requirements. They will assume that all of the screen elements and variables the code refers to exist, and that the code is the working code from your video.

**Row 6 - Response 2C:** The points for Row 6 of the scoring guidelines are awarded strictly for the code segment selected.

Criteria	Decision Rules	Scoring Notes
<p><b>Selected</b> code segment implements an algorithm that <b>includes</b> at least two or more algorithms.</p> <p><b>AND</b></p> <p>At least one of the <b>included</b> algorithms uses mathematical or logical concepts.</p> <p><b>AND</b></p> <p>Explains how one of the <b>included</b> algorithms functions independently.</p>	<p>Responses are still eligible to earn this row, even if they do not earn row 5. The <b>included</b> algorithms can be sub-parts of the algorithm in row 5.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>● the <b>selected</b> algorithm consists of a single instruction;</li> <li>● the <b>selected</b> algorithm consists solely of library calls to existing language functionality;</li> <li>● neither of the <b>included</b> algorithms nor the <b>selected</b> algorithm that includes two or more algorithms uses mathematical or logical concepts;</li> <li>● the code segment consisting of the algorithm is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>● the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).</li> </ul>	<ul style="list-style-type: none"> <li>● Algorithms make use of sequencing, selection or iteration.</li> <li>● Mathematical concepts include mathematical expressions using arithmetic operators and mathematical functions.</li> <li>● Logical concepts include Boolean algebra and compound expressions.</li> <li>● Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.</li> <li>● Selection uses a Boolean condition to determine which of two parts of an algorithm is used.</li> </ul>

**You be the AP Reader!** You are the AP reader trying to determine if they get the point for Row 6. Assume each algorithm snippet below was submitted as part of written response 2c. For each, select yes or no - should the point be awarded, and explain why. (It doesn't have to be a written explanation. You can draw an arrow or circle something in the code with a brief word or label about why.)

<p><b>Example Algorithm 1</b></p> <pre> 1  onEvent("button1", "click", function() { 2    setProperty("screen1", "background-color", "blue"); 3  }); 4  onEvent("id", "click", function(event) { 5    setProperty("screen1", "background-color", "red"); 6  }); </pre>	<p><b>Earn Point? Yes / No</b></p> <p><b>Why?</b></p>
<p><b>Example Algorithm 2</b></p> <pre> 1  onEvent("button1", "click", function() { 2    setProperty("label1", "background-color", "blue"); 3    setProperty("label1", "font-size", randomNumber(20, 40)); 4    playSound("sound://default.mp3", false); 5  }); </pre>	<p><b>Earn Point? Yes / No</b></p> <p><b>Why?</b></p>
<p><b>Example Algorithm 3</b></p> <pre> 1  function manageLogIn() { 2    if (loginStatus=="loggedIn") { 3      showHomeScreen(); 4    } else { 5      resetLogIn(); 6    } 7  } 8  function showHomeScreen() { 9    setScreen("homeScreen"); 10   setText("id", "text"); 11 } 12 function resetLogIn() { 13   loginStatus = "loggedOut"; 14   setScreen("signInScreen"); 15 } </pre>	<p><b>Earn Point? Yes / No</b></p> <p><b>Why?</b></p>

#### Example Algorithm 4

```
1 function increaseScore(){
2   score = score + 1;
3   checkIfEndGame();
4   if(endGame){
5     setText("timeText", "Time:" + totalTime + " secs!")
6   } else {
7     updateScreen();
8   }
9 }
10
11 function updateScreen(){
12   setPosition("button1",randomNumber(0,320), 100);
13   setText("scoreText", score);
14   playSound("sound://default.mp3", false);
15 }
16
17 function checkIfEndGame(){
18   var endTime = getTime();
19   totalTime = (startTime - endTime)/1000;
20   endGame = false;
21   if(score > 10 || totalTime > 20){
22     endGame=true;
23   }
24 }
```

Earn Point? Yes / No

Why?

#### Example Algorithm 5

```
1 while (done == false){
2   var n = promptNum("Enter the secret number");
3
4   if(n == 97){
5     done = true;
6     console.log("You did it! Begin countdown!");
7     for(var i=97; i>=0; i--){
8       console.log(i);
9     }
10    console.log("You're in!");
11  } else {
12    console.log("You've guessed incorrectly");
13    if (n < 97){
14      console.log("Guess higher");
15    } else {
16      console.log("Guess lower");
17    }
18  }
19 }
```

Earn Point? Yes / No

Why?



# Abstraction on the Create PT (20 mins)

## Is it a Good Abstraction?

**Abstraction - College Board Definitions** The AP course framework includes the following statements related to abstraction and programming that are relevant for the Create PT:

- *Multiple levels of abstraction are used to write programs (EU 2.2)*
- *The process of developing an abstraction involves removing detail and generalizing functionality. (EK 2.2.1A)*
- *(Student can) Use abstraction to manage complexity in programs. (LO 5.3.1 [P3])*

**What it means:** Abstractions manage complexity. Programming abstractions make it easier to write complex programs. The AP reader is checking that you can **create, identify, and describe an abstraction** that manages complexity in your code.

- **Something You Wrote:** The code you submit as your abstraction should be something that you wrote entirely on your own. You cannot submit code that a partner helped you write.
- **Good Abstraction Choice - Functions that you wrote:** A function that you wrote means a function that you defined, named, and wrote code for (i.e code that starts `function myFunc() { ... }`). Your function can demonstrate *managing complexity* if it either (or both):
  - gets called from multiple different places in your code
  - has a parameter that generalizes some behavior (and also probably called from several places)
- **Bad Abstraction Choice: `onEvent` and other built-in programming language/environment features**
  - `onEvent` does NOT count as a “student-developed abstraction”. `onEvent` is provided by the programming environment and the code contained within it is used in a single location in your code. Therefore it does not in any way manage complexity.
  - Identifying `onEvent` as an abstraction is the most common way to **lose** credit on this question.
  - **Variables** by themselves are not a data abstraction
  - Anything that is an existing abstraction provided by the language that you are simply using. For example: loops and logical structures are not student developed abstractions



`onEvent` **IS NOT** an abstraction

## Does it Count? - Abstraction Edition

The AP reader has to judge strictly from the code you include in your response to 2d whether it meets the requirements. They will assume that all of the screen elements and variables the code refers to exist, and that the code is the working code from your video.

**Row 7 - Response 2D:** The points for Row 7 of the scoring guidelines are awarded strictly for the code segment selected.

Criteria	Decision Rules	Scoring Notes
<ul style="list-style-type: none"> <li>• Selected code segment is a student-developed abstraction.</li> </ul>	<p>Responses that use existing abstractions to create a new abstraction, such as creating a list to represent a collection (e.g., a classroom, an inventory), would earn this point.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• the response is an <b>existing</b> abstraction such as variables, existing control structures, event handlers, APIs;</li> <li>• the code segment consisting of the abstraction is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>• the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly</li> </ul>	<ul style="list-style-type: none"> <li>• The following are examples of abstractions (EK 5.3.1):                     <ul style="list-style-type: none"> <li>○ Procedures</li> <li>○ Parameters</li> <li>○ Lists</li> <li>○ Application program interfaces (APIs)</li> </ul> </li> <li>• Libraries</li> <li>• Lists and other collections can be treated as abstract data types (ADTs) in developing programs. (EK 5.5.11)</li> </ul>

identifying the code segment containing the abstraction).

**Evaluate Abstractions!** Each of the code segments below shows a portion of code with a rectangle around it. Sometimes additional code is included to help you understand the context of that abstraction (for example, to determine whether the abstraction helps manage complexity).

For each example below respond to 3 things:

- **Earn Point? Yes / No** - Would the selected code segment earn the point for row 7?
- **Why?** - note why it does or doesn't earn the point
- **Manages Complexity? Yes / No** - based on what you can see, are you able to argue that the abstraction manages complexity?

<p><b>Example 1</b></p> <pre>1   onEvent("btn1", "click", function(event) { 2       setText("btn1", "I'm touched."); 3       setProperty("btn1", "font-size", 100); 4       showElement("lbl1"); 5   });</pre>	<p><b>Earn Point? Yes / No</b></p> <p><b>Why?</b></p> <p><b>Manages Complexity? Yes / No</b></p>
<p><b>Example 2</b></p> <pre>1   onEvent("btn1", "click", function(event) { 2       if(score &lt; 0){ 3           setScreen("gameOverScreen") 4       } 5   });</pre>	<p><b>Earn Point? Yes / No</b></p> <p><b>Why?</b></p> <p><b>Manages Complexity? Yes / No</b></p>
<p><b>Example 3</b></p>	<p><b>Earn Point? Yes / No</b></p> <p><b>Why?</b></p> <p><b>Manages Complexity? Yes / No</b></p>

```

1  |  onEvent("btn1", "click", function(event) {
2  |      updatePlayerTurn();
3  |      updateScore(-5);
4  |
5  |      if(score < 0){
6  |          setScreen("gameOverScreen")
7  |      }
8  |      else{
9  |          updateBoard();
10 |      }
11 |  });

```

**Example 4**

```

1  |  onEvent("btn1", "click", function(event) {
2  |      setButtonProps();
3  |  });
4  |
5  |  function setButtonProps(){
6  |      setText("btn1", "I'm touched.");
7  |      setProperty("btn1", "font-size", 100);
8  |      showElement("lbl1");
9  |  }

```

Earn Point? Yes / No

Why?

Manages Complexity? Yes / No

**Example 5**

```

1  |  onEvent("btn1", "click", function(event) {
2  |      setButtonProps();
3  |  });
4  |
5  |  onEvent("screen1", "click", function(event) {
6  |      setButtonProps();
7  |  });
8  |
9  |  function setButtonProps(){
10 |      setText("btn1", "I'm touched.");
11 |      setProperty("btn1", "font-size", 100);
12 |      showElement("lbl1");
13 |  }

```

Earn Point? Yes / No

Why?

Manages Complexity? Yes / No

**Example 6**

Earn Point? Yes / No

Why?

```
1  onEvent("btn1", "click", function(event) {
2      updateScore(5);
3  });
4
5  onEvent("screen1", "click", function(event) {
6      updateScore(-3);
7  });
8
9  function updateScore(amt){
10     totalScore += amt;
11     if( totalScore < 0 ){
12         setScreen("gameOverScreen");
13     }
14 }
```

Manages Complexity? Yes / No

## “Narrow it Down” (15 mins)

You should assume that you’re not going to have enough time to complete the “perfect” project for the Create PT. **This is okay because you can get full credit for a programming project that feels incomplete as long as it has working elements.** While a large or more complete project is satisfying, your score is based mostly on the written responses, which is how you demonstrate that you understand the concepts covered on the Create PT.

All your project actually needs to include is:

- One working feature that you can demonstrate in your video
- An algorithm
- An abstraction

You will make the project much easier if you narrow down your original idea into a simpler target project. This will give you more time to complete the written responses or make smaller improvements.

**How to Narrow it Down:** Narrowing it down means identifying the sub-tasks or sub-problems that meet the PT requirements. Here’s a few strategies to help you do that:

- **Get to the Algorithm:** Split your project into individual components that will likely require an algorithm that meets the Create PT requirements. Each of these components individually could be your entire project.
- **Pick One Part of a Bigger Idea:** Think about your project as a set of programming tasks that each solve part of a larger problem. Some of these individual tasks might suffice for the Create PT. You can just pick one part of a big idea that meets the requirements that you think you can program in the time allotted.
- **Minimal Design Mode - looks don’t matter:** Complex visual design work in Design Mode (setting colors, fonts, spacing, etc.) will likely NOT meet any of the requirements for the Create PT. Don’t worry about how your app looks until *after* you already have code that will let you complete the written responses.

### Practice Narrowing it Down

Below are three descriptions of potential projects that another CS Principles student is considering. For each write:

- Two or three ways they could narrow down the project using the tips above
- Opportunities to write an algorithm in their project even after it’s been narrowed down.

#### Project 1: Tic-Tac-Toe

“Here’s my idea: I want to build a tic-tac-toe game. The user creates an account if they don’t already have one and are taken to the main game board. From there the player will play against the computer in either easy, intermediate, or advanced mode, so I will need to write the code for the computer player. When the game is over their lifetime win total is updated. I will also keep track of how long the game took.”

Ways to narrow down the project (2 or 3)	Algorithm opportunities

### Project 2: Health App

"I volunteer at my local health clinic so I want to build a health app. The user can record information about what they eat, how much they sleep, how much they exercise, and information like their blood pressure and weight. Based on the information provided the app will provide recommendations to the user about how they can improve their health for both diet and exercise. Users can also personalize the look of the app with different theme colors."

Ways to narrow down the project (2 or 3)	Algorithm opportunities

### Project 3: Sports Stats

"I think that I'll build an app that allows the user to quickly record stats during a basketball game. The app will show a picture of the court. The user taps on the court to indicate something happened there. They are presented with a quick menu of options like: shot attempt, foul, steal, rebound, etc. then they select from another list which player did it. At the end of the game it displays a stat sheet for all of the players and the stats for that game."

Ways to narrow down the project (2 or 3)	Algorithm opportunities

## Bring it All Together (15 mins)

With an understanding of the major components of the Create PT, you are ready to start brainstorming projects. While you have at least 12 class hours to complete the task, keep in mind that those 12 hours also must account for time to make your video and complete the written responses. We recommend budgeting at least 5 hours to complete the video and written responses, and so it is highly recommended that you prepare to do a project in which the programming / coding can be completed in 6-7 hours. You want projects with the following features.

- **Personally Relevant:** Pick projects you are actually interested in building.
- **Clear Purpose:** Aim for a simple program whose purpose can be stated in one sentence. For example:
  - The purpose of my program is \_\_\_\_\_.
  - My app does \_\_\_\_\_.
  - My program lets a user \_\_\_\_\_.
  - My app is a \_\_\_\_\_.
- **Narrowed Down:** Repeat the “Narrow it Down” process with your own ideas. A good rule of thumb is that you’ll want to be able to have a first draft of your algorithm within two hours of starting to program.
- **No New Programming Skills:** Make sure you *already* have the programming skills necessary to complete the project. Be flexible. With some creativity you can likely use the skills you’ve already learned to make many different types of projects. Avoid taking on new programming environments or concepts as part of the Create PT.

## Brainstorm Project Ideas

Brainstorm one or two project ideas for the Create PT. Afterwards you’ll share ideas with a classmate for feedback.

Project Idea	Classmate Feedback
<p><i>Purpose:</i></p> <p><i>Ways to Narrow it Down:</i></p> <p><i>Algorithm Opportunities:</i></p> <p><i>Confidence you have skills to do this it in time allotted?:</i></p>	<p><i>Use the list above to give feedback on the idea.</i></p>
<p><i>Purpose:</i></p> <p><i>Ways to Narrow it Down:</i></p> <p><i>Algorithm Opportunities:</i></p> <p><i>Confidence you have skills to do this it in time allotted?:</i></p>	<p><i>Use the list above to give feedback on the idea.</i></p>

## Create PT Progress / Check-in organizer

2a

Program **purpose**  
(remember: your video should show this)

Row 1

2b

Development **process** overall

Row 2

Difficulty/opportunity #1  
Circle one: feedback · testing · reflection  
(how it was resolved, incorporated into project)

Row 3

Difficulty/opportunity #2  
Circle one: feedback · testing · reflection  
(how it was resolved, incorporated into project)

Row 3

2c

Identify main **algorithm** (name of function, location in code, etc.)

Rows 4, 5

How does it function? (explain any  
mathematical or logical concepts used).

Row 5

How does it relate to overall purpose?

Row 5

Algorithm Includes two or more algorithms....

Sub-algorithm 1

Sub-algorithm 2

Explain how at least one of the two (1) uses Mathematical or Logical  
concepts (2) can explain how it functions independently.

Row 6

2d

Your developed **abstraction**  
(name of function(s) you wrote)

How does it manage complexity?

Examples:

- Able to reuse functionality?
- Problem broken down into functions and sub-functions?
- Generalizes specific behavior?

Rows 7,8

This organizer is inspired by the work of Jill Westerlund at [abstractingCS.com](http://abstractingCS.com). Recreated and modified with permission.



# Create PT Completion Timeline

Before you start, you should think about how you are going to allocate your time for the 12 hours provided for the task. Below is a sample timeline that you can use to plan out how you will complete the Create Performance Task.

Hour	Suggested Activity	Your Plan
1 - 2	<p>Begin building a program for a project you brainstormed. Carefully monitor whether you will finish enough of your project in time.</p> <p>Write down one opportunity or difficulty you've encountered for prompt <b>2b</b>.</p> <p><b>Goal:</b> you should be confident after this first round of development that you'll be able to meet the requirements for algorithms (2c) and abstraction (2d).</p> <p><b>Use the <i>Create PT Progress / Check-in organizer</i> to test yourself about whether you are on track.</b></p>	
3 - 4	<p>Keep working. Check in after hour 4 once again on whether you are on track to complete responses. You should ideally know:</p> <ul style="list-style-type: none"> <li>• The abstraction you will write about</li> <li>• The algorithm you will write about</li> </ul> <p>Write down a second opportunity or difficulty you've encountered for prompt <b>2b</b>.</p> <p><b>Use the <i>Create PT Progress / Check-in organizer</i> to test yourself about whether you are on track.</b></p>	
5 - 7	<p>Finalize all programming. After this point you shouldn't be writing more code (beyond simple touch ups)</p>	
8	<p>Complete prompt <b>2b</b>, using the notes you took as you were programming.</p>	
9	<p>Record <b>video</b> of your program running and complete response <b>2a</b></p>	
10	<p>Complete <b>2c</b> describing your algorithm</p>	
11	<p>Complete <b>2d</b> describing your abstraction</p>	
12	<p>Complete the task</p> <ul style="list-style-type: none"> <li>• Review the submission materials</li> <li>• Check your responses against the scoring guidelines</li> <li>• Upload your video, written response, and program code to the digital portfolio</li> </ul> <p><b>Goal:</b> At the end of this day, your Create PT is submitted!</p>	

**Note:** The timeline above is just a guideline. You may complete the performance task on a different schedule. Make sure to leave enough time to complete your computational artifact and write-up.

## Create PT Guidelines

**Video** Submit one video in .mp4, .wmv, .avi, or .mov format that demonstrates the running of at least one significant feature of your program. Your video must not exceed 1 minute in length and must not exceed 30MB in size

**Prompt 2a.** Provide a written response or audio narration in your video that:

- identifies the programming language;
- identifies the purpose of your program; and
- explains what the video illustrates.

*(Must not exceed 150 words)*

**Advice:** For resources on how to make your video, head to <https://studio.code.org/s/csp-create-2019/stage/1/puzzle/2>. Here are the most important things to remember for your video and prompt 2a.

- **Video Runs Continuously:** Your video must run continuously and show your actual code running. It can't just be a series of screenshots.
- **Show One Feature:** Your program does NOT need to be complete so long as you can demonstrate one major feature that's running.
- **Describe the Purpose:** The purpose of your program is the intended goal or objective of the program. In other words, it's "what" the program is supposed to do. If you made a game, an app, or some other kind of project, just quickly describe "what" kind of program it is and how it would be used / played.
- **Connection to Video:** Make sure that you can connect the purpose of your program to what is shown in the video. If you only have one feature working then describe the purpose of the feature.

### Response 2a Checklist

#### Video

- Video runs continuously (it cannot be a series of screenshots)
- Video is less than 60 seconds long and less than 30MB in size
- Video demonstrates one running feature of the program

#### Written Response

- Response identifies the programming language used
- Identifies the purpose of the program
- Describes the feature(s) shown in the video and their connection to the purpose of the program
- May be audio commentary in your video. Carefully follow this checklist even if you use audio commentary.

**2b.** Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and / or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development.  
*(Must not exceed 200 words)*

**Advice:** There are *many* individual pieces of information you need to fit in this response. Use the checklist at the bottom carefully to make sure you don't miss any. Here are the most important pieces to remember:

- **Don't Forget the Overall Process:** You should reference your development process as a whole, NOT just two points in time. Your response should reflect an iterative process of identifying and solving problems focusing on programming decisions (i.e. writing a function to display score) vs design decisions (i.e. how to layout a screen).
- **Difficulties / Opportunities:** You need to describe *two* distinct difficulties / opportunities you encountered
  - A **difficulty** is likely either a bug in your code or a difficult design problem you needed to work out.
  - An **opportunity** is likely an idea or realization you had as you developed your code that led to new

ideas.

- **Feedback / Testing / Reflection:** You should clearly describe HOW you identified the two difficulties / opportunities - was it through testing your program out? Personal reflection? Through feedback from a peer?
- **Independent or Collaborative:** If you developed your program completely independently, you need to say so - don't assume the reader will know. If you developed your program collaboratively, some parts need to be done on your own. For this response *make sure*:
  - You clearly indicate which parts were done collaboratively
  - No identifying information is in your response. It's ok to say "I worked with a partner on this part of the program...". Don't say "I worked with Jesse S to create this part of the program...."
  - You clearly state which of the difficulties/opportunities described here was done *independently on your own* (could be both, but at least one).

## Response 2b Checklist

### Overall Development

- Response describes the *overall* development process, *not only* two key points.
- Response indicates whether you completed the project independently or with a partner. (note: this indication can be incorporated throughout your response *and* in comments within your code as well).

### First Difficulty / Opportunity

- Response describes one difficulty / opportunity encountered early in the development process
- Response describes source of difficulty / opportunity as either feedback, testing, or reflection
- Response indicates how it was incorporated / solved, including whether you wrote the code independently.

### Second Difficulty / Opportunity

- Response describes one difficulty / opportunity encountered later in the development process
- Response describes source of difficulty / opportunity as either feedback, testing, or reflection
- Response indicates how it was incorporated / solved, including whether you wrote the code independently.
- If first Difficulty / Opportunity WAS NOT solved independently, then this one must be

2c. Capture and paste a program code segment that implements an algorithm (marked with an **oval** in **section 3** below) and that is fundamental for your program to achieve its intended purpose. This code segment must be an algorithm you developed individually on your own, must include two or more algorithms, and must integrate mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program. (*Must not exceed 200 words*)

**Advice:** Review the "Is it a Good Algorithm" section above for lots of helpful tips on how to choose your algorithm. Here's the most important points.

- **You Wrote it:** You need to have written the code of your algorithm entirely on your own (not with a partner)
- **Copy and Paste it:** You must paste your actual algorithm code as part of this response.
- **A Parent and Two Children:** Your **selected** algorithm (the "parent") needs to have two **included** algorithms (the "children"). See Example Algorithms 4 and 5 for ideas about how this might look.
- **Mathematical / Logic Concepts:** At least one **included** algorithm needs to use mathematical and/or logical concepts.
- **Break Into Functions:** To make it easier to refer to individual parts of your algorithm give the **selected** and **included** algorithms their own functions. Example Algorithm 4 is written in this way.
- **Describe "how", not just "what":** You need to talk about how your code works, not just what the user will see when it runs. Do this by referring to the actual variables names, programming constructs, strings, and so on, that are visible in your code snippet. For example:

*"The algorithm I selected is `signInUser()` which handles the user login process in my app which has two key parts: `checkName()` and `startHomeScreen()`. `checkName` has an*

*if-statement that checks to see whether the name entered in the usernameTxt textbox is equal to 'MrSillyMan' or 'MsFunnyGal'. If it is then it sets the accessGranted variable to true, otherwise false. The startHomeScreen() function checks the accessGranted variable and returns the login screen if false, otherwise it proceeds to show the home screen for the user."*

### Response 2c Checklist

#### Overall

- You wrote all algorithm code yourself
- Response includes copy-pasted versions of code for main and sub-algorithms with ovals around them
- Response identifies the **selected** algorithm (parent) and at least two **included** algorithms (children).

#### Included algorithm 1

- Clearly identifies the code for the algorithm (where in the code, function name, line numbers, etc)
- Explains what the algorithm does independently
- Describes how the code of the algorithm works
- Uses mathematical or logical concepts

#### Included algorithm 2

- Clearly identifies the code for the algorithm (where in the code, function name, line numbers, etc).
- Explains what the algorithm does independently
- Describes how the code of the algorithm works
- Uses mathematical or logical concepts

#### Selected Algorithm

- Clearly identifies the code for the **selected** algorithm (where in the code, function name, line numbers, etc).
- Describes how **selected** algorithm combines **included** algorithms.
- Explains how **selected** algorithm helps to achieve the overall purpose of the program

**2d.** Capture and paste a program code segment that contains an abstraction you developed individually on your own (marked with a **rectangle** in **section 3** below). This abstraction must integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. (*Must not exceed 200 words*)

**Advice:** Review the “*Is it a Good Abstraction?*” section above for tips on how to choose your abstraction. Here’s the most important points.

- **Choose a Function:** Unless you feel confident about another abstraction, choose a function - not an onEvent, but a function you defined and named yourself.
- **You Wrote it:** You need to have written the code of your abstraction entirely on your own (not with a partner)
- **Copy and Paste it:** You must paste your actual abstraction code as part of this question submission.
- **Manages Complexity:** Make sure you can describe how your abstraction helps manage complexity in your program.
- **Make a contrasting argument:** explain how your program would be *more* complex to read, write, or reason about if you had *not* created your abstraction.
- **Mathematical and Logical Concepts:** While you should aim to include these in your abstraction, this is NOT explicitly assessed by the Scoring Guidelines for 2019.

### Response 2d Checklist

#### Overall

- You wrote all abstraction code yourself (it's not an onEvent block, but a function you defined and named)
- Response includes copy-pasted versions of code for abstraction with a rectangle around it
- Response identifies the abstraction by name
- You explicitly describe HOW the abstraction manages complexity (e.g. by explaining how your code would be more complex to write or reason about without the abstraction)

### 3. Program Code

Capture and paste your entire program code in this section.

- › Mark with an oval the segment of program code that implements the algorithm you created for your program that integrates other algorithms and integrates mathematical and/or logical concepts.
- › Mark with a rectangle the segment of program code that represents an abstraction you developed.
- › Include comments or acknowledgments for program code that has been written by someone else.

**Advice:** For resources on how to make a PDF of your program code head to <https://studio.code.org/s/csp-create-2019/stage/1/puzzle/2>. Here's the most important things to remember:

- **Making Your PDF:** Use [CodePrint](#) to make a PDF of your program. It's designed specifically for the Create PT. You can find it from the link above.
- **Marking Your Algorithm:** Make sure you place an oval around all parts of your algorithm (**selected and included**).
- **Marking Your Abstraction:** Place your rectangle around the code where you create your abstraction (e.g. define a function) not where you use the abstraction (e.g. call a function).
- **Commenting and Collaboration:** You may work with a partner on the Create PT, but you must clearly indicate which parts you completed independently and which you completed together by using comments. For example:

```
// I completed the section below with my collaborative partner
// I completed this section independently
// I have extended code found at [URL]. The code below is my additions
```

Remember that your algorithm and abstraction need to be created entirely independently.

- **Citing Images:** If you use code or images made by someone else (for example that you found online) then you should cite those resources as well. Again you can use comments.

```
// The images used in this app came from:
// [1] bird image - http://name-of-site.com/path/to/image.jpg
// [2] flower image - http://site.com/path/to/flower.jpg
```

# **Unit 7 Lesson 3**

**Create PT: Complete the Task (12 hrs)**

**Resources**

## Create PT Overview

**Goal of the Task:** Create a programming project of your own design and then explain the **purpose, process, algorithms and abstractions** used to build it. You have **12 hours** to complete this task.

**What you Submit:** (1) Video of running program (2) Written Responses to prompts 2a-d (3) PDF of program code

**How you get a good score:** The AP committee wants to see that you can:

- Design and write the code for a computer program with a topic of your choosing
- Describe how you identified and solved problems as you developed your program
- Write code for an algorithm and describe its purpose within your program
- Write code for an abstraction and describe its purpose within your program

**Suggested Process in a Nutshell** (see also the Sample Timeline on following pages):

**Pick your project...** Something small enough that you can design and write in only a few hours.  
You should basically know ahead of time what the core algorithm will be.

**Hours 1-2**      **Develop a “good enough” program / prototype within 2 hours**

- Evaluate whether you can finish what you set out to do, and adjust as necessary.
- Check progress for responses 2c and 2d - algorithms and abstraction.
- You should **know** what **your algorithm and abstraction** will be at this point.

**Hours 3-7**      **Keep coding and get to a stopping point with ~5 hours to go**

- Your video and written responses will take time to make — you’ll want to make last minute improvements!

**Hour 8**      **Record your video and respond to 2a**

**Hours 9-10**      **Write responses to 2b, 2c, 2d**

**Hours 11-12**      **Prepare to submit**

- Finalize program code and written responses and submit on the digital portfolio as three separate files.
  - Video of running program (.mp4, .wmv, .avi, or .mov)
  - Written Response to prompts 2a-2d (PDF)
  - Program Code (PDF)

<sup>1</sup> Much of the content of this this guide was inspired by Jill Westerlund at the [Abstracting CS](#) blog. We are grateful for Jill’s ingenuity and generosity.

# Algorithms on the Create PT (20 mins)

## Is it a Good Algorithm?

**Algorithm - College Board Definitions:** Algorithms are precise sequences of instructions for processes that can be executed by a computer and are implemented using programming languages. Algorithms make use of sequencing, selection or iteration. People write programs to execute algorithms. Your algorithm must include two or more algorithms that work in combination to achieve some desired result. Finally it must integrate mathematical and/or logical concepts.

**What it means:** Algorithms are chunks of code that accomplish a task. To make sure your algorithm is a little complex and that you demonstrate your programming abilities the College Board has a few specific requirements.

- **Something You Wrote:** The entirety of the code you submit as your algorithm should be something that you wrote entirely on your own. You cannot submit code that a partner helped you write.
- **Mathematical and/or Logical Concepts:** Mathematical concepts means code where your app does some sort of mathematical computation (+, -, \*, /, %). These are typically used when your program is making a calculation.

Logical concepts means code where you use logical or comparison operators (&&, ||, !, ==, !=, <, > <= >=). These are typically used in combination with if-statements where your program is making a decision.

- **A Parent and Two Children:** Your **selected** algorithm must have two **included** algorithms that work in combination to achieve some result. The best way to think about this is that you have a “parent” algorithm that requires two or more components that can stand on their own as independent “child” algorithms.

You should NOT submit three separate algorithms. Instead you should find one large task in your program that can actually be divided into two or more sub-tasks (for example: a large task might be making a user login screen; subtasks are (1) checking their login information (2) updating the screen appropriately). The code to accomplish each sub-task are the **included** “child” algorithms that can work independently to solve each sub-task, but are combined to work together as the **selected** “parent” algorithm to solve the overall task.

- **Talking About Your Algorithm:** It is recommended that you place your **selected** algorithm and **included** algorithms in separate functions. This will make it easier for you to talk about your algorithms in your responses. You may also, however, simply place rectangles around them and then use line numbers to identify your algorithms.

## Does it Count? - Algorithm Edition

The AP reader has to judge strictly from the code you include in your response to 2c whether it meets the requirements. They will assume that all of the screen elements and variables the code refers to exist, and that the code is the working code from your video.

**Row 6 - Response 2C:** The points for Row 6 of the scoring guidelines are awarded strictly for the code segment selected.

Criteria	Decision Rules	Scoring Notes
<p><b>Selected</b> code segment implements an algorithm that <b>includes</b> at least two or more algorithms.</p> <p><b>AND</b></p> <p>At least one of the <b>included</b> algorithms uses mathematical or logical concepts.</p> <p><b>AND</b></p> <p>Explains how one of the <b>included</b> algorithms functions independently.</p>	<p>Responses are still eligible to earn this row, even if they do not earn row 5. The <b>included</b> algorithms can be sub-parts of the algorithm in row 5.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>● the <b>selected</b> algorithm consists of a single instruction;</li> <li>● the <b>selected</b> algorithm consists solely of library calls to existing language functionality;</li> <li>● neither of the <b>included</b> algorithms nor the <b>selected</b> algorithm that includes two or more algorithms uses mathematical or logical concepts;</li> <li>● the code segment consisting of the algorithm is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>● the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).</li> </ul>	<ul style="list-style-type: none"> <li>● Algorithms make use of sequencing, selection or iteration.</li> <li>● Mathematical concepts include mathematical expressions using arithmetic operators and mathematical functions.</li> <li>● Logical concepts include Boolean algebra and compound expressions.</li> <li>● Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times.</li> <li>● Selection uses a Boolean condition to determine which of two parts of an algorithm is used.</li> </ul>



**You be the AP Reader!** You are the AP reader trying to determine if they get the point for Row 6. Assume each algorithm snippet below was submitted as part of written response 2c. For each, select yes or no - should the point be awarded, and explain why. (It doesn't have to be a written explanation. You can draw an arrow or circle something in the code with a brief word or label about why.)

<p><b>Example Algorithm 1</b></p> <pre> 1  onEvent("button1", "click", function() { 2    setProperty("screen1", "background-color", "blue"); 3  }); 4  onEvent("id", "click", function(event) { 5    setProperty("screen1", "background-color", "red"); 6  }); </pre>	<p><b>Earn Point? Yes / No</b></p> <p><b>Why?</b></p>
<p><b>Example Algorithm 2</b></p> <pre> 1  onEvent("button1", "click", function() { 2    setProperty("label1", "background-color", "blue"); 3    setProperty("label1", "font-size", randomNumber(20, 40)); 4    playSound("sound://default.mp3", false); 5  }); </pre>	<p><b>Earn Point? Yes / No</b></p> <p><b>Why?</b></p>
<p><b>Example Algorithm 3</b></p> <pre> 1  function manageLogIn() { 2    if (loginStatus=="loggedIn") { 3      showHomeScreen(); 4    } else { 5      resetLogIn(); 6    } 7  } 8  function showHomeScreen() { 9    setScreen("homeScreen"); 10   setText("id", "text"); 11 } 12 function resetLogIn() { 13   loginStatus = "loggedOut"; 14   setScreen("signInScreen"); 15 } </pre>	<p><b>Earn Point? Yes / No</b></p> <p><b>Why?</b></p>

#### Example Algorithm 4

```
1 function increaseScore(){
2   score = score + 1;
3   checkIfEndGame();
4   if(endGame){
5     setText("timeText", "Time:" + totalTime + " secs!")
6   } else {
7     updateScreen();
8   }
9 }
10
11 function updateScreen(){
12   setPosition("button1",randomNumber(0,320), 100);
13   setText("scoreText", score);
14   playSound("sound://default.mp3", false);
15 }
16
17 function checkIfEndGame(){
18   var endTime = getTime();
19   totalTime = (startTime - endTime)/1000;
20   endGame = false;
21   if(score > 10 || totalTime > 20){
22     endGame=true;
23   }
24 }
```

Earn Point? Yes / No

Why?

#### Example Algorithm 5

```
1 while (done == false){
2   var n = promptNum("Enter the secret number");
3
4   if(n == 97){
5     done = true;
6     console.log("You did it! Begin countdown!");
7     for(var i=97; i>=0; i--){
8       console.log(i);
9     }
10    console.log("You're in!");
11  } else {
12    console.log("You've guessed incorrectly");
13    if (n < 97){
14      console.log("Guess higher");
15    } else {
16      console.log("Guess lower");
17    }
18  }
19 }
```

Earn Point? Yes / No

Why?

# Abstraction on the Create PT (20 mins)

## Is it a Good Abstraction?

**Abstraction - College Board Definitions** The AP course framework includes the following statements related to abstraction and programming that are relevant for the Create PT:

- *Multiple levels of abstraction are used to write programs (EU 2.2)*
- *The process of developing an abstraction involves removing detail and generalizing functionality. (EK 2.2.1A)*
- *(Student can) Use abstraction to manage complexity in programs. (LO 5.3.1 [P3])*

**What it means:** Abstractions manage complexity. Programming abstractions make it easier to write complex programs. The AP reader is checking that you can **create, identify, and describe an abstraction** that manages complexity in your code.

- **Something You Wrote:** The code you submit as your abstraction should be something that you wrote entirely on your own. You cannot submit code that a partner helped you write.
- **Good Abstraction Choice - Functions that you wrote:** A function that you wrote means a function that you defined, named, and wrote code for (i.e code that starts `function myFunc() { ... }`). Your function can demonstrate *managing complexity* if it either (or both):
  - gets called from multiple different places in your code
  - has a parameter that generalizes some behavior (and also probably called from several places)
- **Bad Abstraction Choice: `onEvent` and other built-in programming language/environment features**
  - `onEvent` does NOT count as a “student-developed abstraction”. `onEvent` is provided by the programming environment and the code contained within it is used in a single location in your code. Therefore it does not in any way manage complexity.
  - Identifying `onEvent` as an abstraction is the most common way to **lose** credit on this question.
  - **Variables** by themselves are not a data abstraction
  - Anything that is an existing abstraction provided by the language that you are simply using. For example: loops and logical structures are not student developed abstractions



`onEvent` IS NOT an abstraction

## Does it Count? - Abstraction Edition

The AP reader has to judge strictly from the code you include in your response to 2d whether it meets the requirements. They will assume that all of the screen elements and variables the code refers to exist, and that the code is the working code from your video.

**Row 7 - Response 2D:** The points for Row 7 of the scoring guidelines are awarded strictly for the code segment selected.

Criteria	Decision Rules	Scoring Notes
<ul style="list-style-type: none"> <li>• Selected code segment is a student-developed abstraction.</li> </ul>	<p>Responses that use existing abstractions to create a new abstraction, such as creating a list to represent a collection (e.g., a classroom, an inventory), would earn this point.</p> <p><b>Do NOT award a point if any one of the following is true:</b></p> <ul style="list-style-type: none"> <li>• the response is an <b>existing</b> abstraction such as variables, existing control structures, event handlers, APIs;</li> <li>• the code segment consisting of the abstraction is not included in the written responses section or is not explicitly identified in the program code section; or</li> <li>• the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly</li> </ul>	<ul style="list-style-type: none"> <li>• The following are examples of abstractions (EK 5.3.1):                     <ul style="list-style-type: none"> <li>○ Procedures</li> <li>○ Parameters</li> <li>○ Lists</li> <li>○ Application program interfaces (APIs)</li> </ul> </li> <li>• Libraries</li> <li>• Lists and other collections can be treated as abstract data types (ADTs) in developing programs. (EK 5.5.11)</li> </ul>

identifying the code segment containing the abstraction).

**Evaluate Abstractions!** Each of the code segments below shows a portion of code with a rectangle around it. Sometimes additional code is included to help you understand the context of that abstraction (for example, to determine whether the abstraction helps manage complexity).

For each example below respond to 3 things:

- **Earn Point? Yes / No** - Would the selected code segment earn the point for row 7?
- **Why?** - note why it does or doesn't earn the point
- **Manages Complexity? Yes / No** - based on what you can see, are you able to argue that the abstraction manages complexity?

<p><b>Example 1</b></p> <pre>1   onEvent("btn1", "click", function(event) { 2       setText("btn1", "I'm touched."); 3       setProperty("btn1", "font-size", 100); 4       showElement("lbl1"); 5   });</pre>	<p><b>Earn Point? Yes / No</b></p> <p><b>Why?</b></p> <p><b>Manages Complexity? Yes / No</b></p>
<p><b>Example 2</b></p> <pre>1   onEvent("btn1", "click", function(event) { 2       if(score &lt; 0){ 3           setScreen("gameOverScreen") 4       } 5   });</pre>	<p><b>Earn Point? Yes / No</b></p> <p><b>Why?</b></p> <p><b>Manages Complexity? Yes / No</b></p>
<p><b>Example 3</b></p>	<p><b>Earn Point? Yes / No</b></p> <p><b>Why?</b></p> <p><b>Manages Complexity? Yes / No</b></p>

```

1  |  onEvent("btn1", "click", function(event) {
2  |      updatePlayerTurn();
3  |      updateScore(-5);
4  |
5  |      if(score < 0){
6  |          setScreen("gameOverScreen")
7  |      }
8  |      else{
9  |          updateBoard();
10 |      }
11 |  });

```

**Example 4**

```

1  |  onEvent("btn1", "click", function(event) {
2  |      setButtonProps();
3  |  });
4  |
5  |  function setButtonProps(){
6  |      setText("btn1", "I'm touched.");
7  |      setProperty("btn1", "font-size", 100);
8  |      showElement("lbl1");
9  |  }

```

Earn Point? Yes / No

Why?

Manages Complexity? Yes / No

**Example 5**

```

1  |  onEvent("btn1", "click", function(event) {
2  |      setButtonProps();
3  |  });
4  |
5  |  onEvent("screen1", "click", function(event) {
6  |      setButtonProps();
7  |  });
8  |
9  |  function setButtonProps(){
10 |      setText("btn1", "I'm touched.");
11 |      setProperty("btn1", "font-size", 100);
12 |      showElement("lbl1");
13 |  }

```

Earn Point? Yes / No

Why?

Manages Complexity? Yes / No

**Example 6**

Earn Point? Yes / No

Why?

```
1  onEvent("btn1", "click", function(event) {
2      updateScore(5);
3  });
4
5  onEvent("screen1", "click", function(event) {
6      updateScore(-3);
7  });
8
9  function updateScore(amt){
10     totalScore += amt;
11     if( totalScore < 0 ){
12         setScreen("gameOverScreen");
13     }
14 }
```

Manages Complexity? Yes / No

## “Narrow it Down” (15 mins)

You should assume that you’re not going to have enough time to complete the “perfect” project for the Create PT. **This is okay because you can get full credit for a programming project that feels incomplete as long as it has working elements.** While a large or more complete project is satisfying, your score is based mostly on the written responses, which is how you demonstrate that you understand the concepts covered on the Create PT.

All your project actually needs to include is:

- One working feature that you can demonstrate in your video
- An algorithm
- An abstraction

You will make the project much easier if you narrow down your original idea into a simpler target project. This will give you more time to complete the written responses or make smaller improvements.

**How to Narrow it Down:** Narrowing it down means identifying the sub-tasks or sub-problems that meet the PT requirements. Here’s a few strategies to help you do that:

- **Get to the Algorithm:** Split your project into individual components that will likely require an algorithm that meets the Create PT requirements. Each of these components individually could be your entire project.
- **Pick One Part of a Bigger Idea:** Think about your project as a set of programming tasks that each solve part of a larger problem. Some of these individual tasks might suffice for the Create PT. You can just pick one part of a big idea that meets the requirements that you think you can program in the time allotted.
- **Minimal Design Mode - looks don’t matter:** Complex visual design work in Design Mode (setting colors, fonts, spacing, etc.) will likely NOT meet any of the requirements for the Create PT. Don’t worry about how your app looks until *after* you already have code that will let you complete the written responses.

### Practice Narrowing it Down

Below are three descriptions of potential projects that another CS Principles student is considering. For each write:

- Two or three ways they could narrow down the project using the tips above
- Opportunities to write an algorithm in their project even after it’s been narrowed down.

#### Project 1: Tic-Tac-Toe

“Here’s my idea: I want to build a tic-tac-toe game. The user creates an account if they don’t already have one and are taken to the main game board. From there the player will play against the computer in either easy, intermediate, or advanced mode, so I will need to write the code for the computer player. When the game is over their lifetime win total is updated. I will also keep track of how long the game took.”

Ways to narrow down the project (2 or 3)	Algorithm opportunities

### Project 2: Health App

"I volunteer at my local health clinic so I want to build a health app. The user can record information about what they eat, how much they sleep, how much they exercise, and information like their blood pressure and weight. Based on the information provided the app will provide recommendations to the user about how they can improve their health for both diet and exercise. Users can also personalize the look of the app with different theme colors."

Ways to narrow down the project (2 or 3)	Algorithm opportunities

### Project 3: Sports Stats

"I think that I'll build an app that allows the user to quickly record stats during a basketball game. The app will show a picture of the court. The user taps on the court to indicate something happened there. They are presented with a quick menu of options like: shot attempt, foul, steal, rebound, etc. then they select from another list which player did it. At the end of the game it displays a stat sheet for all of the players and the stats for that game."

Ways to narrow down the project (2 or 3)	Algorithm opportunities



## Bring it All Together (15 mins)

With an understanding of the major components of the Create PT, you are ready to start brainstorming projects. While you have at least 12 class hours to complete the task, keep in mind that those 12 hours also must account for time to make your video and complete the written responses. We recommend budgeting at least 5 hours to complete the video and written responses, and so it is highly recommended that you prepare to do a project in which the programming / coding can be completed in 6-7 hours. You want projects with the following features.

- **Personally Relevant:** Pick projects you are actually interested in building.
- **Clear Purpose:** Aim for a simple program whose purpose can be stated in one sentence. For example:
  - The purpose of my program is \_\_\_\_\_.
  - My app does \_\_\_\_\_.
  - My program lets a user \_\_\_\_\_.
  - My app is a \_\_\_\_\_.
- **Narrowed Down:** Repeat the “Narrow it Down” process with your own ideas. A good rule of thumb is that you’ll want to be able to have a first draft of your algorithm within two hours of starting to program.
- **No New Programming Skills:** Make sure you *already* have the programming skills necessary to complete the project. Be flexible. With some creativity you can likely use the skills you’ve already learned to make many different types of projects. Avoid taking on new programming environments or concepts as part of the Create PT.

## Brainstorm Project Ideas

Brainstorm one or two project ideas for the Create PT. Afterwards you’ll share ideas with a classmate for feedback.

Project Idea	Classmate Feedback
<p><i>Purpose:</i></p> <p><i>Ways to Narrow it Down:</i></p> <p><i>Algorithm Opportunities:</i></p> <p><i>Confidence you have skills to do this it in time allotted?:</i></p>	<p><i>Use the list above to give feedback on the idea.</i></p>
<p><i>Purpose:</i></p> <p><i>Ways to Narrow it Down:</i></p> <p><i>Algorithm Opportunities:</i></p> <p><i>Confidence you have skills to do this it in time allotted?:</i></p>	<p><i>Use the list above to give feedback on the idea.</i></p>

## Create PT Progress / Check-in organizer

2a

Program **purpose**  
(remember: your video should show this)

Row 1

2b

Development **process** overall

Row 2

Difficulty/opportunity #1  
Circle one: feedback · testing · reflection  
(how it was resolved, incorporated into project)

Row 3

Difficulty/opportunity #2  
Circle one: feedback · testing · reflection  
(how it was resolved, incorporated into project)

Row 3

2c

Identify main **algorithm** (name of function, location in code, etc.)

Rows 4, 5

How does it function? (explain any  
mathematical or logical concepts used).

Row 5

How does it relate to overall purpose?

Row 5

Algorithm Includes two or more algorithms....

Sub-algorithm 1

Sub-algorithm 2

Explain how at least one of the two (1) uses Mathematical or Logical  
concepts (2) can explain how it functions independently.

Row 6

2d

Your developed **abstraction**  
(name of function(s) you wrote)

How does it manage complexity?

Examples:

- Able to reuse functionality?
- Problem broken down into functions and sub-functions?
- Generalizes specific behavior?

Rows 7,8

This organizer is inspired by the work of Jill Westerlund at [abstractingCS.com](http://abstractingCS.com). Recreated and modified with permission.

# Create PT Completion Timeline

Before you start, you should think about how you are going to allocate your time for the 12 hours provided for the task. Below is a sample timeline that you can use to plan out how you will complete the Create Performance Task.

Hour	Suggested Activity	Your Plan
1 - 2	<p>Begin building a program for a project you brainstormed. Carefully monitor whether you will finish enough of your project in time.</p> <p>Write down one opportunity or difficulty you've encountered for prompt <b>2b</b>.</p> <p><b>Goal:</b> you should be confident after this first round of development that you'll be able to meet the requirements for algorithms (2c) and abstraction (2d).</p> <p><b>Use the <i>Create PT Progress / Check-in organizer</i> to test yourself about whether you are on track.</b></p>	
3 - 4	<p>Keep working. Check in after hour 4 once again on whether you are on track to complete responses. You should ideally know:</p> <ul style="list-style-type: none"> <li>• The abstraction you will write about</li> <li>• The algorithm you will write about</li> </ul> <p>Write down a second opportunity or difficulty you've encountered for prompt <b>2b</b>.</p> <p><b>Use the <i>Create PT Progress / Check-in organizer</i> to test yourself about whether you are on track.</b></p>	
5 - 7	<p>Finalize all programming. After this point you shouldn't be writing more code (beyond simple touch ups)</p>	
8	<p>Complete prompt <b>2b</b>, using the notes you took as you were programming.</p>	
9	<p>Record <b>video</b> of your program running and complete response <b>2a</b></p>	
10	<p>Complete <b>2c</b> describing your algorithm</p>	
11	<p>Complete <b>2d</b> describing your abstraction</p>	
12	<p>Complete the task</p> <ul style="list-style-type: none"> <li>• Review the submission materials</li> <li>• Check your responses against the scoring guidelines</li> <li>• Upload your video, written response, and program code to the digital portfolio</li> </ul> <p><b>Goal:</b> At the end of this day, your Create PT is submitted!</p>	

**Note:** The timeline above is just a guideline. You may complete the performance task on a different schedule. Make sure to leave enough time to complete your computational artifact and write-up.

# Create PT Guidelines

**Video** Submit one video in .mp4, .wmv, .avi, or .mov format that demonstrates the running of at least one significant feature of your program. Your video must not exceed 1 minute in length and must not exceed 30MB in size

**Prompt 2a.** Provide a written response or audio narration in your video that:

- identifies the programming language;
- identifies the purpose of your program; and
- explains what the video illustrates.

*(Must not exceed 150 words)*

**Advice:** For resources on how to make your video, head to <https://studio.code.org/s/csp-create-2019/stage/1/puzzle/2>. Here are the most important things to remember for your video and prompt 2a.

- **Video Runs Continuously:** Your video must run continuously and show your actual code running. It can't just be a series of screenshots.
- **Show One Feature:** Your program does NOT need to be complete so long as you can demonstrate one major feature that's running.
- **Describe the Purpose:** The purpose of your program is the intended goal or objective of the program. In other words, it's "what" the program is supposed to do. If you made a game, an app, or some other kind of project, just quickly describe "what" kind of program it is and how it would be used / played.
- **Connection to Video:** Make sure that you can connect the purpose of your program to what is shown in the video. If you only have one feature working then describe the purpose of the feature.

## Response 2a Checklist

### Video

- Video runs continuously (it cannot be a series of screenshots)
- Video is less than 60 seconds long and less than 30MB in size
- Video demonstrates one running feature of the program

### Written Response

- Response identifies the programming language used
- Identifies the purpose of the program
- Describes the feature(s) shown in the video and their connection to the purpose of the program
- May be audio commentary in your video. Carefully follow this checklist even if you use audio commentary.

**2b.** Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and / or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development.  
*(Must not exceed 200 words)*

**Advice:** There are *many* individual pieces of information you need to fit in this response. Use the checklist at the bottom carefully to make sure you don't miss any. Here are the most important pieces to remember:

- **Don't Forget the Overall Process:** You should reference your development process as a whole, NOT just two points in time. Your response should reflect an iterative process of identifying and solving problems focusing on programming decisions (i.e. writing a function to display score) vs design decisions (i.e. how to layout a screen).
- **Difficulties / Opportunities:** You need to describe *two* distinct difficulties / opportunities you encountered
  - A **difficulty** is likely either a bug in your code or a difficult design problem you needed to work out.
  - An **opportunity** is likely an idea or realization you had as you developed your code that led to new

ideas.

- **Feedback / Testing / Reflection:** You should clearly describe HOW you identified the two difficulties / opportunities - was it through testing your program out? Personal reflection? Through feedback from a peer?
- **Independent or Collaborative:** If you developed your program completely independently, you need to say so - don't assume the reader will know. If you developed your program collaboratively, some parts need to be done on your own. For this response *make sure*:
  - You clearly indicate which parts were done collaboratively
  - No identifying information is in your response. It's ok to say "I worked with a partner on this part of the program...". Don't say "I worked with Jesse S to create this part of the program...."
  - You clearly state which of the difficulties/opportunities described here was done *independently on your own* (could be both, but at least one).

## Response 2b Checklist

### Overall Development

- Response describes the *overall* development process, *not only* two key points.
- Response indicates whether you completed the project independently or with a partner. (note: this indication can be incorporated throughout your response *and* in comments within your code as well).

### First Difficulty / Opportunity

- Response describes one difficulty / opportunity encountered early in the development process
- Response describes source of difficulty / opportunity as either feedback, testing, or reflection
- Response indicates how it was incorporated / solved, including whether you wrote the code independently.

### Second Difficulty / Opportunity

- Response describes one difficulty / opportunity encountered later in the development process
- Response describes source of difficulty / opportunity as either feedback, testing, or reflection
- Response indicates how it was incorporated / solved, including whether you wrote the code independently.
- If first Difficulty / Opportunity WAS NOT solved independently, then this one must be

2c. Capture and paste a program code segment that implements an algorithm (marked with an **oval** in **section 3** below) and that is fundamental for your program to achieve its intended purpose. This code segment must be an algorithm you developed individually on your own, must include two or more algorithms, and must integrate mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program. (*Must not exceed 200 words*)

**Advice:** Review the "Is it a Good Algorithm" section above for lots of helpful tips on how to choose your algorithm. Here's the most important points.

- **You Wrote it:** You need to have written the code of your algorithm entirely on your own (not with a partner)
- **Copy and Paste it:** You must paste your actual algorithm code as part of this response.
- **A Parent and Two Children:** Your **selected** algorithm (the "parent") needs to have two **included** algorithms (the "children"). See Example Algorithms 4 and 5 for ideas about how this might look.
- **Mathematical / Logic Concepts:** At least one **included** algorithm needs to use mathematical and/or logical concepts.
- **Break Into Functions:** To make it easier to refer to individual parts of your algorithm give the **selected** and **included** algorithms their own functions. Example Algorithm 4 is written in this way.
- **Describe "how", not just "what":** You need to talk about how your code works, not just what the user will see when it runs. Do this by referring to the actual variables names, programming constructs, strings, and so on, that are visible in your code snippet. For example:

*"The algorithm I selected is `signInUser()` which handles the user login process in my app which has two key parts: `checkName()` and `startHomeScreen()`. `checkName` has an*

*if-statement that checks to see whether the name entered in the usernameTxt textbox is equal to 'MrSillyMan' or 'MsFunnyGal'. If it is then it sets the accessGranted variable to true, otherwise false. The startHomeScreen() function checks the accessGranted variable and returns the login screen if false, otherwise it proceeds to show the home screen for the user."*

### Response 2c Checklist

#### Overall

- You wrote all algorithm code yourself
- Response includes copy-pasted versions of code for main and sub-algorithms with ovals around them
- Response identifies the **selected** algorithm (parent) and at least two **included** algorithms (children).

#### Included algorithm 1

- Clearly identifies the code for the algorithm (where in the code, function name, line numbers, etc)
- Explains what the algorithm does independently
- Describes how the code of the algorithm works
- Uses mathematical or logical concepts

#### Included algorithm 2

- Clearly identifies the code for the algorithm (where in the code, function name, line numbers, etc).
- Explains what the algorithm does independently
- Describes how the code of the algorithm works
- Uses mathematical or logical concepts

#### Selected Algorithm

- Clearly identifies the code for the **selected** algorithm (where in the code, function name, line numbers, etc).
- Describes how **selected** algorithm combines **included** algorithms.
- Explains how **selected** algorithm helps to achieve the overall purpose of the program

**2d.** Capture and paste a program code segment that contains an abstraction you developed individually on your own (marked with a **rectangle** in **section 3** below). This abstraction must integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. (*Must not exceed 200 words*)

**Advice:** Review the “*Is it a Good Abstraction?*” section above for tips on how to choose your abstraction. Here’s the most important points.

- **Choose a Function:** Unless you feel confident about another abstraction, choose a function - not an onEvent, but a function you defined and named yourself.
- **You Wrote it:** You need to have written the code of your abstraction entirely on your own (not with a partner)
- **Copy and Paste it:** You must paste your actual abstraction code as part of this question submission.
- **Manages Complexity:** Make sure you can describe how your abstraction helps manage complexity in your program.
- **Make a contrasting argument:** explain how your program would be *more* complex to read, write, or reason about if you had *not* created your abstraction.
- **Mathematical and Logical Concepts:** While you should aim to include these in your abstraction, this is NOT explicitly assessed by the Scoring Guidelines for 2019.

### Response 2d Checklist

#### Overall

- You wrote all abstraction code yourself (it's not an onEvent block, but a function you defined and named)
- Response includes copy-pasted versions of code for abstraction with a rectangle around it
- Response identifies the abstraction by name
- You explicitly describe HOW the abstraction manages complexity (e.g. by explaining how your code would be more complex to write or reason about without the abstraction)

### 3. Program Code

Capture and paste your entire program code in this section.

- › Mark with an oval the segment of program code that implements the algorithm you created for your program that integrates other algorithms and integrates mathematical and/or logical concepts.
- › Mark with a rectangle the segment of program code that represents an abstraction you developed.
- › Include comments or acknowledgments for program code that has been written by someone else.

**Advice:** For resources on how to make a PDF of your program code head to <https://studio.code.org/s/csp-create-2019/stage/1/puzzle/2>. Here's the most important things to remember:

- **Making Your PDF:** Use [CodePrint](#) to make a PDF of your program. It's designed specifically for the Create PT. You can find it from the link above.
- **Marking Your Algorithm:** Make sure you place an oval around all parts of your algorithm (**selected and included**).
- **Marking Your Abstraction:** Place your rectangle around the code where you create your abstraction (e.g. define a function) not where you use the abstraction (e.g. call a function).
- **Commenting and Collaboration:** You may work with a partner on the Create PT, but you must clearly indicate which parts you completed independently and which you completed together by using comments. For example:

```
// I completed the section below with my collaborative partner
// I completed this section independently
// I have extended code found at [URL]. The code below is my additions
```

Remember that your algorithm and abstraction need to be created entirely independently.

- **Citing Images:** If you use code or images made by someone else (for example that you found online) then you should cite those resources as well. Again you can use comments.

```
// The images used in this app came from:
// [1] bird image - http://name-of-site.com/path/to/image.jpg
// [2] flower image - http://site.com/path/to/flower.jpg
```