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 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 “child” algorithms that can work independently to solve each sub-task, but can are combined to work together as a “parent” algorithm to solve the overall task.

- **Talking About Your Algorithm:** It is recommended that you place your main 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
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 independently.	**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).	● Algorithms make use of sequencing, selection or iteration. ● Mathematical concepts include mathematical expressions using arithmetic operators and mathematical functions. ● Logical concepts include Boolean algebra and compound expressions. ● Iteration is the repetition of part of an algorithm until a condition is met or for a specified number of times. ● Selection uses a Boolean condition to determine which of two parts of an algorithm is used.

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:

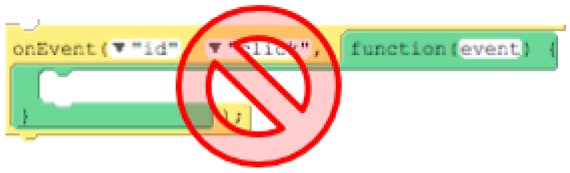
- *Multiple levels of abstraction are used to write programs (EU 2.2)*
- *The process of developing an abstraction involves removing detail and generalizing functionality. (EK 2.2.1A)*
- *(Student can) Use abstraction to manage complexity in programs. (LO 5.3.1 [P3])*

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("id", "click", function(event) {
});
```

*`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.

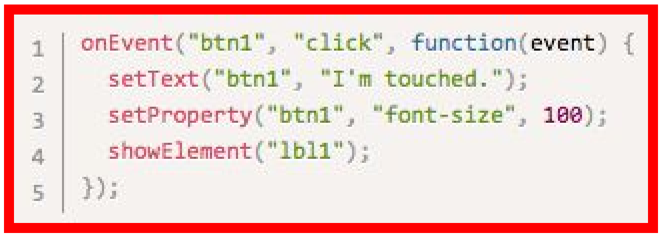
Criteria	Decision Rules	Scoring Notes
• Selected code segment is a student-developed abstraction.	Responses that use existing abstractions to create a new abstraction, such as creating a list to represent a collection (e.g., a classroom, an inventory), would earn this point. **Do NOT award a point if 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	• The following are examples of abstractions (EK 5.3.1): ○ Procedures ○ Parameters ○ Lists ○ Application program interfaces (APIs) • Libraries • Lists and other collections can be treated as abstract data types (ADTs) in developing programs. (EK 5.5.1I)

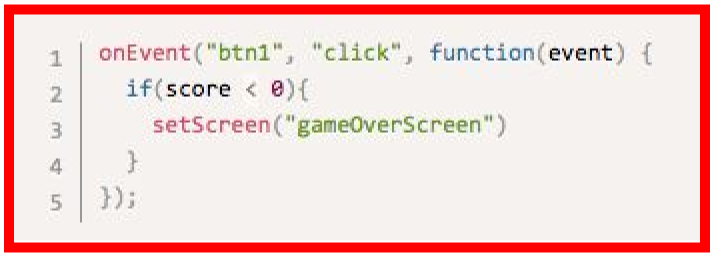
	identifying the code segment containing the abstraction).	

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?

Example 1  `onEvent("btn1", "click", function(event) {`
`  setText("btn1", "I'm touched.");`
`  setProperty("btn1", "font-size", 100);`
`  showElement("lbl1");`
`});`
	Earn Point? **Yes / No**
	Why?
	Manages Complexity? **Yes / No**

Example 2  `onEvent("btn1", "click", function(event) {`
`  if(score < 0){`
`    setScreen("gameOverScreen")`
`  }`
`});`
	Earn Point? **Yes / No**
	Why?
	Manages Complexity? **Yes / No**

Example 3	**Earn Point?** **Yes / No** **Why?** **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 ok 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

Written Response Templates

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/stage/1/puzzle/2. Here's 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.

Sentence Starters
- *My program is written in JavaScript using the App Lab programming environment.*
- *The purpose of my program is… (describe what it was meant to do)*
- *In the video you can see (how a piece of functionality works that's directly tied to the purpose).*

Draft Your Response Here:

(must not exceed 150 words)

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's the most important pieces to remember:

- **Don't Forget the Overall Process:** Your response should reference your development process as a whole, NOT just the two points in time. Your response should reflect an iterative process of identifying and solving problems.
- **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.
- **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
  - You clearly state which of the difficulties/opportunities described here was done *independently on your own* (could be both, but at least one).

Sentence Starters
- *I completed this project (independently / collaboratively).*
- *I began developing my program by (describe beginning of process, what did you do / decide first)*
- *Then I worked iteratively to (how you continued building your project)*
- *Early on through (feedback / testing / reflection) I identified a (problem / opportunity) which was …*
- *I solved this problem (independently / with my partner) by …*

Draft Your Response Here:

(must not exceed 200 words)

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 main algorithm (the "parent") needs to have two sub-algorithms (the "children"). See Example Algorithms 4 and 5 for ideas on how this might look.
- **Mathematical / Logic Concepts:** At least one sub-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 parent and child 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."

Sentence Starters
- *The main algorithm that I selected is (main-algorithm name).*
- *This algorithm has two key parts, (sub-algorithm 1) and (sub-algorithm 2).*
- *(Sub-algorithm 1) is designed to (what it does). It does this by (how the code of sub-algorithm 1 works).*
- *My (main-algorithm) combines (sub-algorithm 1) and (sub-algorithm 2) to (what main-algorithm does).*
- *Together these algorithms help achieve the purpose of my program by (how these algorithms are tied to program purpose).*

Draft Your Response Here:

[don't forget - paste the algorithm code snippet here - the same one you put an oval around in the whole program code]

[write your response]

(must not exceed 250 words)

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 main algorithm and at least two sub-algorithms

Sub-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

Sub-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

Main Algorithm

- ❏ Clearly identifies the code for the algorithm (where in the code, function name, line numbers, etc).
- ❏ Describes how main algorithm combines sub-algorithms
- ❏ Explains how main 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 2018.

Sentence Starters
- *The abstraction I selected is (name of abstraction)*
- *My abstraction manages complexity in my program by…*
- *Without my abstraction my program would be more difficult to (read / write / understand) because...*

Draft Your Response Here:

[Don't forget - paste your abstraction code snippet here - the same one you put a rectangle around in the program code]

[Write your response]

(must not exceed 250 words)

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)