Chief Reader Report on Student Responses:

2022 AP® Computer Science Principles Free-Response Questions

- Number of Students Scored 134,651
- Number of Readers 403
- Score Distribution

Exam Score	N	%At
5	15,322	11.4
4	28,249	21.0
3	41,931	31.1
2	26,799	19.9
1	22,350	16.6

- Global Mean 2.91

The following comments on the 2022 free-response questions for AP® Computer Science Principles were written by the Chief Reader, Tom Cortina, Carnegie Mellon University. They give an overview of each free-response question and of how students performed on the question, including typical student errors. General comments regarding the skills and content that students frequently have the most problems with are included. Some suggestions for improving student preparation in these areas are also provided. Teachers are encouraged to attend a College Board workshop to learn strategies for improving student performance in specific areas.

Question 1

Task: Create Performance Task
Topic: Application from Ideas

	Max. Points:	Mean Score:
Row 1	1	0.47
Row 2	1	0.64
Row 3	1	0.33
Row 4	1	0.46
Row 5	1	0.43
Row 6	1	0.37

What were the responses to this question expected to demonstrate?

The Create Performance Task is designed to give students an opportunity to develop a program that solves a problem for the user or allows the pursuit of a creative interest. Students should be able to demonstrate the program running in a short video and explain its purpose, how it functions, and how it handles input and output of information as shown in the video.

Programs typically process collections of data to help the user gain insight and make decisions. This task also requires students to demonstrate their understanding of data abstraction, using at least one list (or equivalent collection type) to hold data that is critical to fulfilling the program's purpose. Students must explain how the list manages complexity in the program, by either explaining why their program could not function without the list or why their program would require a more complex implementation without the use of the list, to demonstrate the importance of using this abstraction when processing larger amounts of data.

Programs use procedures to break a larger computational task into smaller subtasks to make a program easier to develop and test. This task also requires students to use procedural abstraction to write a procedure with at least one explicit parameter that demonstrates the use of sequencing, selection, and iteration, along with a call to this procedure. The student should be able to explain how the procedure works in detail, what the procedure does in summary, and how the procedure contributes to the overall functionality of the program. Finally, the student should be able to explain how to test the procedure for correctness using its parameter(s), using two examples that cause different behavior and results to occur.

How well did the responses address the course content related to this question? How well did the responses integrate the skills required on this question?

Program Purpose and Function

- Students were asked to develop a working program with a purpose to solve a problem for the user or pursue a creative interest. Students were also asked to create a video that demonstrated some functionality of their program that requires user input and program output. In general, most students were able to write a working program and create a video that demonstrated the program's functionality, input, and output. In addition, most students were able to accurately describe the functionality, input, and output in their written response.

- Some students were able to state the purpose of the program by describing a broader problem that is being solved for the user or the creative, artistic interest being explored, whereas other students incorrectly stated the functionality as the purpose. In the case of game programs, some students incorrectly stated the object of the game (i.e., how to win) as the purpose of the program. Some students did not describe what was being illustrated in their video.

Data Abstraction and Managing Complexity

- Students were asked to provide two code segments from their program, the first showing the initialization of a list of data and the second showing how that list is used in their program to reduce complexity. Some responses used other acceptable representations of lists, including arrays and dictionaries (associative arrays). Most students were able to identify a list in their program and provide two code segments that demonstrated the initialization and use of the identified list. Some students confused creating an empty list with initializing the list with data. Some students accessed only one specific element of the list instead of multiple elements, although accessing a single random element of the list was allowed.

- Some students were able to use the lists in a manner that reduces complexity by generalizing list element access using an index or subscript to allow for arbitrary-sized lists and iteration over lists. In general, these students were able to explain in their written response how the list managed complexity in their program. On the other hand, some students wrote about managing complexity with lists in general and did not describe how their own program managed complexity with a list. Some students inaccurately described that their program could not be completed without lists when the use of the list was simplistic and could be replaced with a few individual variables.

Procedural Abstraction, Algorithm Implementation, and Testing

- Students were asked to provide two code segments from their program, the first showing a procedure using at least one explicit parameter required to perform its function and the second which shows a call to this procedure with argument(s) for the parameter(s). Students were also asked to write about what the procedure does and how it contributes to the overall program. Students who described what the procedure does at a higher level, rather than giving a line-by-line explanation, found it easier to connect the procedure to its contribution to the program. Most students were able to provide the two code segments and describe what the procedure does. Some students had trouble explaining how the procedure contributed to the overall program, particularly when the selected procedure performed almost the entirety of the overall functionality of the program. Some students submitted procedures with no explicit parameters (e.g., using global variables to "pass" data to the procedure) or code fragments not enclosed in a procedure. When

students submit multiple procedures, the score is based on the first procedure and those that it calls. Some students were confused about the requirement of two code segments and submitted two procedures that weren't related.

- Students were asked to demonstrate sequencing, selection, and iteration in their code segment. While the requirements state that the algorithm should be included in a procedure, an exception is made to allow students to earn credit for demonstrating their ability to write an algorithm with sequencing, selection, and iteration even if it isn't included in a procedure with parameters. Additionally, students were asked to write a description of how their code segment works in enough detail so that someone else could recreate it. In general, most students were able to present a code segment that used sequencing, selection, and iteration, but some failed to explain the code in enough detail to allow someone else to write a similar code segment. Some students used a built-in `timedLoop` function as an iteration instead of using a built-in iteration command with a loop condition.

- Students were asked to describe two calls to the code segment representing their procedure, with each call passing a different argument(s), explicit or implicit, that cause a different segment of code to execute in their procedure. In each case, students were asked to describe what condition was being tested by each call and what results from each call. Some students were able to describe different argument(s) that caused separate paths to execute in the procedure, resulting in unique results. On the other hand, some students described calls to two different procedures, or they described calls to the same procedure with different arguments and different results but that followed the same sequence of instructions in the procedure. While the requirements state that the algorithm should have one or more parameters explicitly shown, an exception is made to allow students to earn credit if the response uses an implicit parameter to simulate the passing of an argument to a procedure by setting a global variable to be used by the procedure just prior to calling the procedure.

What common student misconceptions or gaps in knowledge were seen in the responses to this question?

Common Misconceptions/Knowledge Gaps	*Responses that Demonstrate Understanding*
Video and Written Response 3a: Program Purpose and Function	
Row 1 • Confusing purpose and function when describing the program illustrated in the video component. The function of the program is not its purpose. For example, "The purpose of the program is to display statistics (rank, % of World Population that speaks the language) about a language the user inputs in the `languageInput` box" describes the function of the program by describing what it does, rather than the purpose, which should explain why someone would use the program. If the program is a game, the purpose is not the object of the game. For example, "the purpose of this program is to find out how many turtles the user can load into the elevator in 15 seconds" describes the object of the game, rather than the purpose, which would address why someone would use the program.	Row 1 • High-scoring responses stated a purpose for the program that went beyond the function of the program itself. These responses indicated how the program would be used to solve a problem for the user or improve their lives in some way; that is, a reason why someone would use the program. For example, "a major problem in healthcare today is that patient files can easily be lost or mixed up, so I made a program with the overall purpose to solve the problem if a mix up were to happen."
Written Response 3b: Data Abstraction and Managing Complexity	
Row 2 • Providing several lists and not showing how either list is being used. For example, a student may provide two code segments, one that initializes `animalImages` and one that initializes `animalList` (containing names of the animals). However, neither code segment shows how these data values are being used in the program. • Only accessing the length of the list instead of elements within the list. For example, score = captureList.length.	Row 2 • High-scoring responses identify only one list and show how data is added to that list in the first code segment and how multiple elements of the list are accessed in the second code segment. • High-scoring responses access multiple elements of the list rather than its length. For example, for the list `priceList`, a loop is set up with loop variable `i` that runs from index `0` to `length(priceList) - 1`, and the body of the loop computes `sum += priceList[i]`.

• Not indicating that a list parameter is the same as the identified list. For example, when the list `x` is passed as an argument to the parameter `myl`, the response does not explicitly mention that the two names (`x` and `myl`) refer to the same list.	• High-scoring responses identify in a reasonable manner that when using a procedure to process the list, the list parameter is the same list as the one initialized in the prior code segment. For example, "the specific list I chose is `nounList`. As it is called and used in the procedure `selectWord`, however, it is referred to as the parameter `wordList` instead."
Row 3 • Explaining how the use of the list manages complexity by stating how the program would be rewritten in a generic way that does not reference the submitted program or explaining that writing the program would be impossible without the use of the list even though it is possible to write. For example, "This list manages complexity in the program because if it weren't for the lists, each individual variable would have had to be written out and given its own name." This generic answer applies to all uses of lists and does not address how the program code of the specific program would need to be modified to accommodate all the additional variables. Another example: "If I didn't have a list containing the artist names, I wouldn't be able to show the user who sang the song they are looking for, or even find the artist they are looking for when they input the one they want to find." In this case there are other programming solutions that would allow them to write the program.	Row 3 • High-scoring responses describe how the program would need to be rewritten in a more complex manner without the list, referencing specific instructions and functionality in the submitted program. For example, "Without the list over 110+ different sprites would have to be spawned manually. As an example the code for the tree would look something like this: `Tile((1038,587), [self.visible_sprites, self.obstacle_sprites])` but repeated over 30 times, each time a different number would have to be found out for the x and y coordinates. The cat code would be even more complex as with each instance of a cat there is a 50 percent chance of it spawning at that location, so that piece of code along with the code that draws the cat would have to be repeated over 79 times!"

• Using a list in a manner that does not manage complexity, such as:
o Using a list with only two elements in it, which could be replaced with two variables.
o Using a list of strings that are only output in the same order as they are stored in the list, which could be replaced with a single string using concatenation.
o Using a list as a counter, storing arbitrary data in the list just to determine its length. For example, storing five copies of the number 1 in a list and then reporting the score is 5 because the length of the list is 5.
	• High-scoring responses display lists that are used to manage complexity by storing a variable number of data values that need to be processed using a loop or iterator. For example,
	`for (var j = 0; j < ltrs.length; j++)`
	`{`
	`  if (letter == ltrs[j])`
	`  {`
	`  setText("let" + j, letter);`
	`  check = true;`
	`  count = count + 1;`
	`  }`
	`}`

Written Response 3c: Procedural Abstraction and Algorithm Implementation	
Row 4	Row 4
• Not including both what the procedure does at a high level AND how it contributes to the functionality of the program. For example, "`startgame()` creates the terminal and begins a 15-second timer." This response does not state how this procedure contributes to the overall program, such as stating when it is used or what triggers it to be executed.	• High-scoring responses include two parts to the response, stating the overall function of the procedure in a sentence and then also stating how it contributes to the overall program by describing when it is executed or what subsequent code is run in the program as a result of this procedure having been executed. For example, "This identified procedure uses selection to test the user input to see which option it equals. This contributes to the functionality of the overall program by deciding which character function to run and call based on user input."

• Including a parameter that has no effect on the procedure either by ignoring the parameter or initializing it to another value. For example,
`def winner(player):`
`  player = 1`
`  ...`
	• High-scoring responses use the parameter value(s) in the procedure body without overwriting the argument being passed into the procedure.

• Failing to use one or more explicit parameters in the submitted procedure, instead "passing" data to the procedure using global variables.	• High-scoring responses include a procedure that has one or more explicit parameters that allow some part of the program to send specific data to the procedure for its use without other parts of the code modifying the data while it is being used.
• Including a larger segment of code that contains multiple procedures; either the procedures aren't related to each other through calls, or the response includes two different procedures as the two code segments without showing a proper procedure call for either one.	• High-scoring responses include one clearly defined procedure with explicit parameters in the first code segment and a single procedure call with relevant argument(s) in the second code segment.
• Including code that is not encapsulated in a procedure.	• High-scoring responses include one clearly defined procedure with explicit parameters in the first code segment.

Row 5
• Including an algorithm that does not contain iteration or have meaningful iteration. Some responses included extensive nested selection (if/else if/else if/…) but no iteration. Some responses included a loop that only ran through one iteration. For example,
`y = 1;`
`for (var x = 1; x <= y; x++) {`
`  total = total + score;`
`}`
	Row 5
	• High-scoring responses included a loop that contributed meaningfully to the algorithm. These loops often repeated a number of times greater than 3 or ran until a specific condition was met (e.g., by using a while loop).

• Using selection in a trivial manner. For example, testing if `players > 0` when `players` is always greater than `0`.	• High-scoring responses included reasonable uses of selection with conditions that were nontrivial and caused different parts of the algorithm to execute.
• Providing a very vague and brief description that does not include enough detail for someone else to recreate the algorithm.	• High-scoring responses included enough detail of each step of the algorithm to allow someone to reasonably recreate the algorithm. Some responses included line numbers that were referenced in the response to make the description easier to follow.
Written Responses 3d: Testing	
Row 6 • Describing calls to two different procedures rather than two calls to the same procedure with different arguments. For example, `move_left(2)` and `move_right(2)`.	Row 6 • High-scoring responses included two calls to the same procedure with different arguments that caused different behavior to occur in the procedure. For example, `move_down(4)` and `move_down(-2)`. (In the case of a negative argument, this procedure returns immediately and does not perform the move.)
• Describing two different parts of the program and what behavior occurs during these parts.	• High-scoring responses identify the single procedure given in response 3c and provide two different sets of values for the parameters that cause different behavior to occur inside the procedure.
• Describing two calls to the same procedure with different arguments that cause the same sequence of code to execute in both cases. A common misconception in this situation is that a procedure that has two different return	• High-scoring responses provided two different arguments that would cause a different sequence of code to execute in each case, leading to a unique result (output or return value). For example, using the function `search(mylist,`

values must be executing different instructions inside. For example, using the function `search(mylist, val)` to find the index of the value in the list, where `mylist = [1,2,3,4,5]` and `val = 1` for the first call and `val = 4` for the second call. Both calls cause the same set of instructions to execute but return different results.	`val)` to find the index of the value in the list, where `mylist = [1,2,3,4,5]` and `val = 1` for the first call and `val = 8` for the second call. Each call causes a different sequence of instructions to execute since in one case, the value is found in the list and in the other case, the value is not found in the list.

• Setting the value of an implicit parameter inside the procedure instead of immediately before the procedure is called. For example, using `choice` as the implicit parameter,
 	
`function select()`
`{`
`  choice = getText("Name");`
`  ...`
`}`
	• High-scoring responses that used implicit parameters set the value of the parameter immediately before the call to the procedure to simulate parameter passing. For example, using `choice` as the implicit parameter,
	
	`choice = getText("Name");`
	`select();`
	`...`
	`function select()`
	`{`
	`  ...`
	`}`

• Identifying input arguments and output values but not describing what conditions are being tested in each case.	• High-scoring responses included a description of each call, stating what code is being tested in each case and how these tests are unique.
• Providing descriptions that were implausible given the code supplied in response 3c. For example, a response indicates that an argument will cause a specific output after a loop is completed, but the loop never executes given the specific argument.	• High-scoring responses accurately described what conditions were being tested for each test case without making statements that were inconsistent or implausible with the submitted procedure.
• Including code that is not encapsulated in a procedure.	• High-scoring responses that addressed two separate testing cases included one clearly defined procedure with explicit or implicit parameters in the first code segment.

Based on your experience at the AP® Reading with student responses, what advice would you offer teachers to help them improve the student performance on the exam?

In general, give students several opportunities to complete a practice Create Performance Task of shorter duration to gain a better understanding of the learning objectives and skills required for the task. Score these against the established scoring criteria to help students improve their understanding. Alternatively, provide several completed tasks for students to review and analyze, asking them to determine whether each requirement was met or not, and why. As a teacher, review the high-quality examples to make sure you understand the nuances of the scoring criteria so that you can score the shorter student examples accurately.

Students need explicit instruction and experience taking screen captures of code segments and incorporating them into their responses. Code submitted for scoring should be as clear as possible (not blurry), and text should be at least 10-point font size.

If a student wishes to use an unconventional programming language for the Create Performance Task, evaluate its ability to clearly address the requirements of the task and advise the student accordingly.

The following bulleted list gives more specific advice for each part of the Create Performance Task.

Responses 2 and 3a: Program Purpose and Function

- Have students review high-quality examples of the Create Performance Task to become familiar with the difference between function and purpose. Give students additional examples of computer programs and ask them what the purpose of each program is to see if they can identify the problem being solved by the program or the creative or artistic pursuit. Ask students to think about "why" the program exists as opposed to "what" the program does or "how" to win the computer game.
- Ensure that students have access and opportunity to practice using computational video tools to capture their program features. Integrate the use of computational tools such as screen capture and creating short videos into multiple assignments. Assist students in learning how to make sure any text in the video is clearly visible and readable for scoring.
- Give students examples of computer programs and ask them to identify what explicit data are being input to the program and what explicit data are being output to the user.
- Make it clear to students that while it is okay to base their program on a program used in class, they must make significant changes to the program by adding additional functionality. The program code used for their written responses should be newly student-developed program code.

Response 3b: Data Abstraction and Managing Complexity

- Give students examples of program code that initializes and uses a list, highlighting the difference between initializing a list and creating an empty list. Have students identify in the code where the initialization and use of the list are happening.
- Give examples that use lists that can be of arbitrary length to illustrate the power of using lists to store a collection of data. Compare these to examples where the lists are of fixed length. Have students write code to process elements of an arbitrary length list by using an index (e.g., `numlist[j]`) rather than hardcoding access (e.g., `numlist[1]`).
- Provide practice using lists with a large number of elements that will require students to write code that is more abstract and able to handle a change in the number of elements more easily than if the code was written in a more hardcoded way with a list containing just a few elements.
- Discuss why a well-designed list makes the code less complex by showing what would happen if the list were not present. Have students explain in their own words why the list is necessary by referencing the code. Show students examples of lists that can be replaced easily without making the code more complex (e.g., a list containing data only to later determine how many items are in the list, which can be replaced with a counter variable).

- Remind students that even if their code segment uses multiple lists, they should clearly identify one list and respond to the prompts based on this one identified list only. The list they identify and describe must be one they create, not one that they are given, such as a data table from a third-party provider.

Response 3c: Procedural Abstraction and Algorithm Implementation

- Give students examples of program code that contain procedures with explicit parameter(s), and have the student identify what code makes up the procedure and where the parameter(s) get their values. Show how using global variables is not the same communication mechanism as using parameters to pass data into a procedure.

- Encourage students to use explicit parameters over implicit parameters since this will make their code easier to debug and easier to explain for the Create Performance Task. Remember that points are not awarded in row 4 if only implicit parameters are used.

- Remind students to identify one procedure with explicit parameter(s) to focus their response and to only include this procedure and a call to this procedure in response 3c. If the procedure calls additional student-developed procedures, students can include them as well in the first code segment, but they should be listed as subsequent to the first procedure, and the student should be sure to focus their response on the first procedure.

Response 3d: Testing

- Give examples of testing a single procedure with different arguments, and trace the path taken in the code to generate each result to show that the paths are different or that different parts of the code are being tested.

- Give an example where a response shows two different procedures, with one test case per procedure, and explain why this response does not address the requirements for the task.

What resources would you recommend to teachers to better prepare their students for the content and skill(s) required on this question?

- The endorsed providers for AP Computer Science Principles have a wealth of resources and examples, including guides, for completing the Create Performance Task. We recommend teachers leverage these resources as instructional tools to help students understand the requirements for the Create Performance Task. When doing so, be sure to clearly communicate to students that when completing their task, they need to write their answers in their own words and avoid modelling their answers too closely to the example responses. Using phrasing from public samples is a violation of the plagiarism policy and will result in students receiving a zero on the task.

- The College Board webinar “Tips for Completing the Create Performance Task—AP CSP” provides guidance on how to break down the Create Performance Task over the course of 12 hours. It also answers many frequently asked student questions. (https://www.youtube.com/watch?v=LfzpMASeNHg)

- The College Board webinar “Score the Create Performance Task, Just Like an AP Reader” provides teachers with insight on how each row of the scoring guidelines is applied to the samples and was conducted in a similar manner to how readers are trained. (https://www.youtube.com/watch?v=mCM3cFBBJvo)