2023

AP®

AP® Computer Science Principles

Sample Student Responses and Scoring Commentary

Inside:

Performance Task—Create

☑ Scoring Guidelines

☑ Scoring Commentary

Student Samples provided separately

Create Performance Task — 6 points

General Scoring Notes

- Responses should be evaluated solely on the rationale provided.
- Responses must demonstrate all criteria, including those within bulleted lists, in each row to earn the point for that row**.**
- Terms and phrases defined in the terminology list are italicized when they first appear in the scoring criteria.

Reporting Category	Scoring Criteria	Decision Rules
Row 1 **Program Purpose and Function** **(0–1 points)**	The video demonstrates the running of the program including: • *input* • *program functionality* • *output* **AND** The written response: • describes the overall *purpose* of the program. • describes what functionality of the program is demonstrated in the video. • describes the input and output of the program demonstrated in the video.	**Consider ONLY the video and written response 3a when scoring this point.** **Do NOT award a point if the following is true:** • The video does not show a demonstration of the program running (screenshots or storyboards are not acceptable and would not be credited). • The described purpose is actually the functionality of the program. The purpose must address the problem being solved or creative interest being pursued through the program. The function is the behavior of a program during execution and is often described by how a user interacts with it.
Row 2 **Data Abstraction** **(0–1 points)**	The written response: • includes two *program code segments*: - one that shows how *data has been stored in this list* (or other *collection type*). - one that shows the data in this same *list being used* as part of fulfilling the program’s purpose. • identifies the name of the variable representing the list being used in this response. • describes what the data contained in this list is representing in the program.	**Consider ONLY written response 3b when scoring this point.** **Requirements for program code segments:** • The written response must include two clearly distinguishable program code segments, but these segments may be disjointed code segments or two parts of a contiguous code segment. • If the written response includes more than two code segments, use the first two code segments to determine whether or not the point is earned. **Do NOT award a point if any one or more of the following is true:** • The list is a one-element list. • The use of the list does not assist in fulfilling the program’s purpose.

Reporting Category	Scoring Criteria	Decision Rules
Row 3 Managing Complexity **(0–1 points)**	The written response: • includes a program code segment that shows a list being used to manage complexity in the program. • explains how the named, selected list manages complexity in the program code by explaining why the program code could not be written, or how it would be written differently, without using this list.	**Consider ONLY written response 3b when scoring this point.** **Responses that do not earn the point in row 2 may still earn the point in this row.** **Do NOT award a point if any one or more of the following is true:** • The code segments containing the lists are not separately included in the written response section (not included at all, or the entire program is selected without explicitly identifying the code segments containing the list). • The written response does not name the selected list (or other collection type). • The use of the list is irrelevant or not used in the program. • The explanation does not apply to the selected list. • The explanation of how the list manages complexity is implausible, inaccurate, or inconsistent with the program. • The solution without the list is implausible, inaccurate, or inconsistent with the program. • The use of the list does not result in a program that is easier to develop, meaning alternatives presented are equally complex or potentially easier. • The use of the list does not result in a program that is easier to maintain, meaning that future changes to the size of the list would cause significant modifications to the code.
Row 4 Procedural Abstraction **(0–1 points)**	The written response: • includes two program code segments: - one showing a *student-developed procedure* with at least one *parameter* that has an effect on the functionality of the procedure. - one showing where the student-developed procedure is being called. • describes what the identified procedure does and how it contributes to the overall functionality of the program.	**Consider ONLY written response 3c when scoring this point.** **Requirements for program code segments:** • The procedure must be student developed but could be developed collaboratively with a partner. • If multiple procedures are included and none are specifically called out in the written response, use the first procedure listed to determine whether the point is earned. • The parameter(s) used in the procedure must be explicit. Explicit parameters are defined in the header of the procedure. **Do NOT award a point if any one or more of the following is true:** • The code segment consisting of the procedure is not included in the written responses section. • The procedure is a built-in or existing procedure or language structure, such as an event handler or main method, where the student only implements the body of the procedure rather than defining the name, return type (if applicable), and parameters. • The written response describes what the procedure does independently without relating it to the overall function of the program.

Reporting Category	Scoring Criteria	Decision Rules
Row 5 **Algorithm Implementation** **(0–1 points)**	The written response: • includes a program code segment of a *student-developed algorithm* that includes: - *sequencing* - *selection* - *iteration* • explains in detailed steps how the identified algorithm works in enough detail that someone else could recreate it.	**Consider ONLY written response 3c when scoring this point.** **Responses that do not earn the point in row 4 may still earn the point in this row.** **Requirements for program code segments:** • The algorithm being described can utilize existing language functionality or library calls. • An algorithm that contains selection and iteration also contains sequencing. • An algorithm containing sequencing, selection, and iteration that is not contained in a procedure can earn this point. • Use the first code segment, as well as any included code for procedures called within this first code segment, to determine whether the point is earned. • If this code segment calls other student-developed procedures, the procedures called from within the identified procedure can be considered when evaluating whether the elements of sequencing, selection, and iteration are present, as long as the code for the called procedures is included. **Do NOT award a point if any one or more of the following is true:** • The response only describes what the selected algorithm does without explaining how it does it. • The description of the algorithm does not match the included program code. • The code segment consisting of the selected algorithm is not included in the written response. • 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 use of either the selection or the iteration is trivial and does not affect the outcome of the program.

Reporting Category	Scoring Criteria	Decision Rules
Row 6 **Testing** **(0–1 points)**	The written response: • describes two calls to the selected procedure identified in written response 3c. Each call must pass a different *argument(s)* that causes a different segment of code in the algorithm to execute. • describes the condition(s) being tested by each call to the procedure. • identifies the result of each call.	**Consider ONLY the written response for 3d and the selected procedure identified in written response 3c.** **Responses that do not earn the point in row 4 may still earn the point in this row.** **Requirements for program code segments:** • Consider implicit or explicit parameters used by the selected procedure when determining whether this point is earned. Implicit parameters are those that are assigned in anticipation of a call to the procedure. For example, an implicit parameter can be set through interaction with a graphical user interface. • A condition that uses the procedure's parameter(s) to execute two different code segments can earn this point. • A condition that uses the procedure's parameter(s) to execute or bypass a code segment can earn this point. **Do NOT award a point if any one or more of the following is true:** • A procedure is not identified in written response 3c. • The written response for 3d does not apply to the procedure in 3c. • The two calls cause the same exact sequence of code in the algorithm to execute even if the result is different. • The two calls are to two different procedures. • The response describes conditions being tested that are implausible, inaccurate, or inconsistent with the program. • The identified results of either call are implausible, inaccurate, or inconsistent with the program.

AP Computer Science Principles Create Performance Task Terminology (in order of appearance in the scoring guidelines)

Input: Program input is data that are sent to a computer for processing by a program. Input can come in a variety of forms, such as tactile (through touch), audible, visual, or text. An event is associated with an action and supplies input data to a program.

Program functionality: The behavior of a program during execution, often described by how a user interacts with it.

Output: Program output is any data that are sent from a program to a device. Program output can come in a variety of forms, such as tactile, audible, visual, movement, or text.

Purpose: The problem being solved or creative interest being pursued through the program.

Program code segment: A code segment refers to a collection of program statements that are part of a program. For text-based, the collection of program statements should be continuous and within the same procedure. For block-based, the collection of program statements should be contained in the same starter block or what is referred to as a "Hat" block.

List: A list is an ordered sequence of elements. The use of lists allows multiple related items to be represented using a single variable. Lists are referred to by different terms, such as arrays or arraylists, depending on the programming language.

Data has been stored in this list: Input into the list can be through an initialization or through some computation on other variables or list elements.

Collection type: Aggregates elements in a single structure. Some examples include: databases, hash tables, dictionaries, sets, or any other type that aggregates elements in a single structure.

List being used: Using a list means the program is creating new data from existing data or accessing multiple elements in the list.

Student-developed procedure / algorithm: Program code that is student developed has been written (individually or collaboratively) by the student who submitted the response. Calls to existing program code or libraries can be included but are not considered student developed. Event handlers are built-in abstractions in some languages and will therefore not be considered student-developed. In some block-based programming languages, event handlers begin with "when".

Procedure: A procedure is a named group of programming instructions that may have parameters and return values. Procedures are referred to by different names, such as method, function, or constructor, depending on the programming language.

Parameter: A parameter is an input variable of a procedure. Explicit parameters are defined in the procedure header. Implicit parameters are those that are assigned in anticipation of a call to the procedure. For example, an implicit parameter can be set through interaction with a graphical user interface.

Algorithm: An algorithm is a finite set of instructions that accomplish a specific task. Every algorithm can be constructed using combinations of sequencing, selection, and iteration.

Sequencing: The application of each step of an algorithm in the order in which the code statements are given.

Selection: Selection determines which parts of an algorithm are executed based on a condition being true or false. The use of try / exception statements is a form of selection statements.

Iteration: Iteration is a repetitive portion of an algorithm. Iteration repeats until a given condition is met or for a specified number of times. The use of recursion is a form of iteration.

Argument(s): The value(s) of the parameter(s) when a procedure is called.

Create Performance Task

Note: Student samples are quoted verbatim and may contain spelling and grammatical errors.

Overview

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.

Create Performance Task (continued)

Sample Identifier: A

Score:
Row 1: 1
Row 2: 1
Row 3: 1
Row 4: 1
Row 5: 1
Row 6: 1

Row 1:
The response earned the point for this row, meeting all six criteria:

- The video demonstrates the running of the program showing input (the initial angle and velocity), program functionality (the projectile tracing a path), and output (the dots showing the trajectory of the projectile).
- The response describes the overall purpose of the program as "to entertain and to educate students by deepening knowledge of projectiles."
- The response describes the functionality demonstrated in the video. The response states, "A realistic projectile simulation is displayed based on user inputs changing the initial angle and velocity. The projectile shoots across the screen, and after it goes off the screen, dots trace in with density showing speed." This description matches what is shown in the video.
- The response describes the input as "The user inputs the up and down arrow" and "The user can also input the right and left arrow." The response describes the output as "a black ball animates across the screen" and "dotted lines with density of lines showing speed."

Row 2:
The response earned the point for this row, meeting all three criteria:

- The response includes two program code segments. The first segment shows data being stored in the list `positions` by appending `pos_tuple` at the end of the update function. The second segment shows multiple elements in the list `positions` being accessed in the for loop.
- The name of the list is identified as `positions`.
- The response describes the data contained in the list as "the position the projectile has been at each frame," representing "every position the object has been at this run."

Create Performance Task (continued)

Row 3:
The response earned the point for this row, meeting both criteria:

- The response includes code that uses a list to manage complexity by storing the positions of the trajectory and iterating over this list to draw the trajectory of the projectile.
- The response explains how the code could not be written without a list. The response states, "without a list, the program would have to somehow create a new variable for each tuple at each frame and create new variables in real time while figuring out how to wait and draw a dot at each of these." The response recognizes that this would be "extremely complex and messy if possible."

Row 4:
The response earned the point for this row, meeting both criteria:

- The response includes a student-developed procedure, `render`, with two parameters, `positions` and `pos_tuple`, that are used in the procedure. The response also includes a call to this procedure in a separate segment of code.
- The response describes what the identified procedure does, stating that the procedure "clears the previous screen by covering it with white. It then displays the angle and velocity before firing. While firing, the procedure draws the circle at each position determined by the update function producing an animated ball movement. Once the ball is off the screen, the procedure also draws a dot at the position the ball was at every fourth frame." The response also describes how the procedure contributes to the overall functionality of the program. The response states, "Displaying the object adds to the functionality by showing the realistic movement of the projectile in a parabolic motion and how changing velocity and angle transform the parabola."

Create Performance Task (continued)

Row 5:
The response earned the point for this row, meeting both criteria:

- The student-developed algorithm within the procedure `render` includes sequencing, selection (if statement), and iteration (for loop).
- The response concisely explains in detail how the algorithm in the procedure works so it can be recreated. The response states, "The algorithm first fills the screen white to clear the screen. Next, the function defines a font variable to use when drawing the velocity and angle. Then, the function coverts the angle in radians represented by variable `ang` to degrees and rounds to the nearest whole degree to account for small errors. The procedure then turns that new degree and the velocity variable into strings so they can be printed to the screen later. After that, the procedure makes two new variables for the font image for the degree and velocity string. Next, the function draws both font images to the screen. After drawing the starting velocity and angle, the function draws a black circle at the position tuple parameter location. After drawing the object, the procedure checks to see if index `1` of the position tuple parameter is greater than `750`. This determines if the ball is close to going of screen. If yes, the procedure iterates over the positions list and draws a dot at every fourth element in the list which are position tuples. If not greater than `750` it does nothing. Finally, the function flips the screen."

Row 6:
The response earned the point for this row, meeting all three criteria:

- The response describes two different calls to the procedure, `render`, resulting in different code being executed. The response gives the first call as "`render(positions, pos_tuple)` when `pos_tuple = (50,50)`," which causes the body of the first if statement not to execute. The response gives the second call as "`render(positions, pos_tuple)` when `pos_tuple = (1000, 760)`," which causes the body of the first if statement to execute.
- The response describes the condition being tested. The response describes both tests as testing "if the `pos_tuple` index `1` is greater than `750`." Because of the values used in the two calls and the different outcomes described in the response, the response overall makes it clear that the two tests test different outcomes of this condition.
- The response identifies the outcome of each call. For the first call, the response states, "the program skips the code under the conditional and does not draw the dots." For the second call, the response states, "the program executes the code under the conditional and iterates over the programs list drawing dots at every fourth position tuple contained in the list."

Create Performance Task (continued)

Sample Identifier: B

Score:
Row 1: 1
Row 2: 1
Row 3: 1
Row 4: 1
Row 5: 1
Row 6: 1

Row 1:
The response earned the point for this row, meeting all six criteria:

- The video demonstrates the running of the program showing input (clicking on the fruits), program functionality (a user playing the fruit clicking game), and output (the fruits appearing and then disappearing in response to the user's clicks, the text messages).
- The response describes the overall purpose of the program as "to entertain the audience with a clicking game."
- The response describes the functionality demonstrated in the video as "it allows the user to gain points and beat other players by clicking on the fruits as they appear randomly as time decreases." The response goes on to describe how the player will lose a life if they click on a grape. All this functionality is shown in the video.
- The response describes the input as "the user-generated answer to the question of which level the user wishes to play at." The response describes the output as "the speed at which the fruits move."

Row 2:
The response earned the point for this row, meeting all three criteria:

- The response includes two program code segments. The first segment shows data being stored at position `1` of the list `leaderboard`. The second segment shows the list being used by sorting it and then accessing element `1`. Sorting the list processes all elements in the list.
- The name of the list is identified as `leaderboard`.
- The response states that the data contained in the list "represents the scores each user obtained through the game."

Create Performance Task (continued)

Row 3:
The response earned the point for this row, meeting both criteria:

- The response includes code that uses a list to manage complexity by storing all the scores from previous runs of the game. This list grows dynamically as more players play the game.
- The response explains how the list manages complexity "by holding the score values in an easily accessible and codable place." The response also describes how the code would be more complex without the list. The response states, "If I did not use a leaderboard to organize the points, I would need to manually write out each score onto the screen and use if-else statements to see if each score is greater than or less than the other values."

Row 4:
The response earned the point for this row, meeting both criteria:

- The response includes a student-developed procedure, `level`, with one parameter, `#`. The parameter is used in the procedure to determine the level of the game. The response also includes a call to this procedure in a separate segment of code.
- The response describes how the procedure contributes to the overall functionality of the program by "allow[ing] for different speeds on the sprites." The response also describes what the procedure does: "If the user inputs easy, the sprites will go to random positions and wait three seconds before moving. If the user inputs hard, the sprites will go to random positions but will wait one second before moving."

Row 5:
The response earned the point for this row, meeting both criteria:

- The student-developed algorithm within the procedure `level` includes sequencing, selection (if statement), and iteration (repeat-until loop).
- The response concisely explains how the algorithm in the procedure works with enough detail that it can be recreated. The response describes the first part of the code—i.e., the if statement, including the loop in the if statement body—as "there is an if else statement with the condition that if the parameter `'#'`, which is changed to the variable `'answer'`, is equal to `easy`, the sprite will go to a random position and then wait three seconds. This will be repeated until the variable `'time'` equals zero." It then describes what happens in the else statement in a similar way.

Create Performance Task (continued)

Row 6:
The response earned the point for this row, meeting all three criteria:

- The response describes two different calls to the procedure, `level`, with two different inputs: “easy difficulty” and “hard difficulty.” The first call will result in the body of the if statement being executed, while the second call will result in the body of the else statement being executed.
- The response describes the condition being tested for each call. The response describes that in the first call “the procedure will run the ‘if’ part of the if else statement,” while in the second call “the procedure will skip the ‘if’ part of the if else statement and run the ‘else’ portion.”
- The response identifies the outcome of each call. For the first call, the response states that the result will be “the sprites waiting for three seconds before going to a random position on the stage,” while for the second call the response states that the result will be “the sprites waiting for one second before going to a random position on the stage.” This description matches what will happen with the program in each case.

Create Performance Task (continued)

Sample Identifier: C

Score:
Row 1: 1
Row 2: 1
Row 3: 1
Row 4: 1
Row 5: 0
Row 6: 1

Row 1:
The response earned the point for this row, meeting all six criteria:

- The video demonstrates the running of the program showing input (text answers to questions), program functionality (selecting a pet based on the user's answers to the questions), and output (the questions for the user and the pet recommendation).
- The response describes the overall purpose of the program: "The program's purpose is to provide the user entertainment and amusement."
- The response describes the functionality demonstrated in the video. The response states, "The program uses the user's inputs in order to ask first if they want to participate, and then four multiple-choice questions about their interests. … The final pet is determined by the number range of the final score." This description matches what is shown in the video.
- The response describes the input as "the user typing their name, then a yes or no depending on if they want to play, and then multiple-choice answers depending on what their interests are." The response describes the output as "the personalized pet generated."

Row 2:
The response earned the point for this row, meeting all three criteria:

- The response includes two program code segments. The first segment shows data being stored in the list `yes_list`. The second segment shows the list being used. As the user's answer is being checked to see if it is contained in `yes_list`, multiple elements in `yes_list` must be accessed.
- The name of the list is identified as `yes_list`.
- The response describes the data contained in the list as "multiple versions of yes the user might input if they want to continue the program."

Create Performance Task (continued)

Row 3:
The response earned the point for this row, meeting both criteria:

- The response includes code that uses a list to manage complexity by storing nine ways that the user might say yes. The list manages complexity by allowing the program to use "in" to check if a given string is a valid way of saying "yes," instead of hard-coding the acceptable "yes" values in one or more conditional statements or storing them in separate variables.
- The response explains how the code would be more complex without the list. The response states, "I could have assigned a variable for each yes possibility, which would require an if-statement for each variable to check for equivalency to the user's answer."

Row 4:
The response earned the point for this row, meeting both criteria:

- The response includes a student-developed procedure, `point_calc`, with one parameter, `input`, that is used in the procedure. The response also includes four calls to this procedure in a separate segment of code. (Only one call is needed to meet this criterion.)
- The response describes how the procedure contributes to the overall functionality of the program. The response states, "The procedure, `point_calc`, adds to the overall functionality because it is integral to output the personalized pet for the user" because "`point_calc` is necessary to … determine the pet." This last statement also describes what the identified procedure does: it "calculate[s] points to determine the pet."

Row 5:
The response did not earn the point for this row, meeting one of the two criteria:

- The student-developed algorithm within the procedure `point_calc` includes sequencing, selection (if statement), and iteration (for loop). However, the use of iteration is trivial. The response states, "In order to increase the range of numbers to determine each pet, I run through the loop multiple times." However, the same effect could have been achieved more simply without the loop by adding 4, 8, 12, and 16 to the variable points in each of the if-statements, respectively, instead of adding 1, 2, 3, and 4, respectively, four times.
- The response concisely explains how the algorithm in the procedure works with enough detail that it can be recreated. The response states, "The procedure, `point_calc`, takes one input as its parameter. It makes the existing variable, `points`, global. Then it evaluates each if-statement four times (due to the "for in range (4)" loop) to see if it applies to the argument. There are four if-statements each for if the argument equals `A`, `B`, `C`, or `D`. For the one it applies to, the assigned number of points is added to `'points'`. Each argument will only apply to one of the if-statements, so when it repeats the other three loops the same number of points will be added each time. There is no need for a return value because `'points'` stores the total points." This response describes its input and global variable, the for loop, and what each of the if statements does. Although this description does not specify how many points to add in each case, it is clear from the response what "the assigned number of points" means.

Create Performance Task (continued)

Row 6:
The response earned the point for this row, meeting all three criteria:

- The response describes two different calls to the procedure, `point_calc`. For the first call, the response states, “In this example, `q1= A` (meaning the user picked A).” That is, the parameter passed to the procedure is `"A"`. For the second call, the response states, “In this example, `q2= B`,” meaning the parameter passed to the procedure is `"B"`. These two calls will result in the bodies of different if statements executing.
- The response describes the condition being tested for each call. For the first call the response states, “The condition being tested is if `q1` is equal to the string in each statement. Since `A` is the value of `q1`, this would activate only the first if-statement.” For the second call the response states, “The condition being tested is if `q2` is equal to the string in each statement. Since `B` is the value of `q2`, this would activate only the second if-statement.”
- The response identifies the outcome of each call. For the first call, the response states, “The result of the parameter `q1` is 4 points stored in `'points'`.” For the second call, the response states, “The result of the parameter `q2` is 8 points stored in `'points'`.”

Create Performance Task (continued)

Sample Identifier: D

Score:
Row 1: 1
Row 2: 1
Row 3: 0
Row 4: 1
Row 5: 1
Row 6: 1

Row 1:
The response earned the point for this row, meeting all six criteria:

- The video demonstrates the running of the program, showing input into the program through pressing keys to navigate through the maze and to destroy obstacles. Output includes the movement of the character. This satisfies the first three criteria for the row.
- The response specifies the program's purpose: "The purpose is to provide a pathway for entertainment for kids 3-7 by providing an interactive maze game that requires the use of timing and decision-making while also being fun."
- The response describes the functionality demonstrated in the video. The response states, "The user utilizes the WASD keys to move around a character while shooting water at balls. The balls are inside a maze that the character must go through without touching anything." Additional functionality is described.
- The response describes the input and the output. Regarding input, the response states, "The inputs shown in the video are: when the green flag is clicked, when you type in either 'easy' or 'hard', when you press 'W',' A',' S',' D' when you press 'space', and lastly when the user-controlled character touches anything within the `'touchList'`." Regarding output, the response states, "The outputs shown in the video are: the `'difficultyAnswer'` block which makes `'diff' = 1` or `'diff' = 2` which activates the `'bossHits'` block and the `'easy/hardSpawn'` block." Additional outputs are listed.

Row 2:
The response earned the point for this row, meeting all three criteria:

- Two distinct code segments are provided, the first showing data being stored in the identified list and the second showing the data being accessed from the identified list as the loop condition.
- The name of the list is identified as `waterHitList`.
- The response identifies what is stored in the list. The response states, "The list contains objects that when hit should hide the water which is constantly moving."

Create Performance Task (continued)

Row 3:
The response did not earn the point for this row, meeting one of the two criteria.

- The response includes code that uses a list to manage complexity by storing the Boolean values that need to be checked as the character moves through the maze.
- The response explains how the code would be written differently without the list. However, the response only provides a generic response regarding adding if statements. The response states, “The checking of these values would have to be in multiple ‘ifs’ that would be constantly repeated thus slowing down the code.” The list given is already a set of conditional tests, so replacing these with individual if statements would not change the complexity of the code.

Row 4:
The response earned the point for this row, meeting both criteria:

- The response includes a student-developed procedure, `difficultyAnswer`, with a parameter that is used in the procedure. The response also includes a call to this procedure.
- The response describes the functionality of the procedure. The response states, “The procedure asks the user for the difficulty they want and broadcasts the answer.” The response describes how the procedure contributes to the overall program by stating it “determines how hard the game is and if the boss fight occurs.”

Row 5:
The response earned the point for this row, meeting both criteria:

- The student-developed algorithm within procedure `difficultyAnswer` includes sequencing, selection (an if-else statement), and iteration (a repeat until loop).
- The response explains how the algorithm works in detail. It states, “First, the algorithm takes a variable called `'reset'` and sets it to `1` which prevents the ball spawning from previous resets to continue. Afterward, it has a ‘repeat until’ which locks the code into an infinite loop with only 2 escapes. It does this by checking if `'repeatQuestion'` is `1` and then checking the difficulty parameter to see if it has either `'easy'` or `'hard'`.” Additional detailed steps are given in the response.

Create Performance Task (continued)

Row 6:
The response earned the point for this row, meeting all three criteria:

- The response describes two different calls to the specific procedure to result in different code being executed. The response identifies the first call as `"difficultyAnswer(easy)"`. The response states that the second call is `"difficultyAnswer(eezy)"`.
- The response describes the conditions being tested. The response states, "This is testing the 'if' part of the 'if-else' statement which goes into another 'if' statement. The first 'if' takes only lets the user through if they had inputted `'easy'` or `'hard'` and the second 'if' takes this and activates the code within either if `'easy'` or if `'hard'`." The response describes the second condition: "This is testing the else part of the if-else statement. This part should repeat if the user doesn't input `'easy'` or `'hard'`."
- The response describes the results of the two calls, leading to two different results. The response states the first result as "The diff is set to `1` and the broadcast is sent resulting in the spawn speed of the balls being 5 seconds, the boss health being `5`, and the `'repeatQuestion'` variable being changed to `1`" and the result of the second call as "'The guy'(user-controlled character) says, 'Please say easy or hard' and asks the question again."

Create Performance Task (continued)

Sample Identifier: E

Score:
Row 1: 1
Row 2: 1
Row 3: 0
Row 4: 1
Row 5: 1
Row 6: 0

Row 1:
The response earned the point for this row, meeting all six criteria:

- The video demonstrates the running of the program, showing input into the program using the keyboard to set the difficulty and clicking on the animals. The output results on the screen as the score and the order in which the animals were clicked.
- The response specifies the program's purpose. The response states, "The overall purpose of the program is to entertain users with a win or lose game of click-on-me."
- The response describes the functionality demonstrated in the video. The response states, "The program functionality shows the program monkey sprite welcoming the user to the game and asking if they want an easy or hard game. We then see the user playing a game of Click On Me with "Easy" settings," winning the game by clicking most times on the monkey. We then see the sprite congratulating the user for winning and showing how many times they clicked on it and the elephant."
- The response describes the input and the output. The response states, "The input shown is the user typing `'1'` or `'2'` for an easy or hard game, typing `'yes'` or `'no'` to play again or not, and clicking on the elephant and monkey sprites." For output, the response states, "The output we see in the video is the sprite introducing itself, asking the user if the user would like to play an easy or hard game, explaining the instructions, displaying whether they won or lost along with the list of times they clicked on either of the sprites, asking if the user wants to play again, and saying thank you for playing when the user is done."

Row 2:
The response earned the point for this row, meeting all three criteria:

- Two code segments are provided. The first segment shows data being stored in the identified list, and a second segment shows data being accessed from the list. The list is displayed as part of the game, identifying the order in which the animals were clicked, which is part of the program's purpose.
- The name of the list is identified as `ElephantOrChimp`.
- The response identifies what is stored in the list. The response states, "The data in `'ElephantOrChimp'` is a list of strings that contains the words `'elephant'` and `'chimp'` based on which animal the user clicks on."

Create Performance Task (continued)

Row 3:
The response did not earn the point for this row, meeting one of the two criteria:

- The response includes a list that collects the type and order of the animal that is clicked so that this sequence can be remembered and displayed at the end of the game.
- The response provides an inaccurate explanation of why the program could not be written differently without the list. The response states, "Without the list there would not be a way of showing the user the pattern in which the chimp and elephant were clicked." The use of the list could be replaced by a string that appends the new data when the animal is clicked.

Row 4:
The response earned the point for this row, meeting both criteria:

- The response includes a student-developed procedure, `playgame`, with one parameter, `EasyHard`, that is used in the procedure.
- The response describes the functionality of the procedure as it states, "The procedure itself is the click-on-me game and, based on the user input, plays the game on an easy or difficult setting." The response describes how the identified procedure contributes to the overall functionality. The response states, "It contributes to the overall functionality of the program because it is the primary source of using the program which is to play the click-on-me game and is called each time the user wishes to play."

Row 5:
The response earned the point for this row, meeting both criteria:

- The student-developed algorithm within procedure, `playgame`, includes sequencing, selection (if-else statement), and iteration (repeat until loop).
- The response explains in detail how the algorithm in the procedure works so it can be recreated. For example, the response states, "The procedure has a parameter called `EasyHard`. If `EasyHard` is equal to `1`, broadcast `beginMoving` Then the sprites will move to a random position and wait three seconds before jumping to a new random position. This repeats until length of the list `ElephantOrChimp` is greater than nine. After that, broadcast `stopMoving` initiates, the sprite moves to x position 0 and y position 0. If `winTimes` is greater than `loseTimes` the sprite will say, 'Congrats you win!' for two seconds. If not the sprite will say, 'Oh too bad, you lost!' for two seconds." More detail is given for the other steps of the algorithm.

Create Performance Task (continued)

Row 6:
The response did not earn the point for this row, meeting two of the three criteria:

- The response correctly describes two calls to the procedure with different arguments. The response states, “The first call is with the number `1`.” The response describes the second call as, “The second call is with the number `2`.”
- The response describes the conditions being tested for the two procedure calls, but one of these conditions is stated incorrectly. The response correctly states the first call is “Testing to see if the first IF section executes which will cause the game to start at an easy mode having the sprites jump to random positions every 3 seconds.” However, the response incorrectly states the second call is “Testing to see if the first IF section executes which will cause the game to start at a hard mode having the sprites jump to random positions every 1 second.” The test should be to see if the second if section executes.
- The response correctly identifies the results of each call leading to different outcomes. The response states that the first call results in “The game is initiated with a difficulty of `'easy'` as the chimp and elephant jump to random positions at a pace of three seconds per move.” The response states that the second call results in “The game is initiated with a difficulty of `'hard'` as the chimp and elephant jump to random positions at a pace of one second per move.”

Create Performance Task (continued)

Sample Identifier: F

Score:
Row 1: 1
Row 2: 1
Row 3: 1
Row 4: 0
Row 5: 1
Row 6: 0

Row 1:
The response earned the point for this row, meeting all six criteria:

- The video demonstrates the running of the program, showing input into the program through keyboard typing. Output includes the title of the suggested reading, and the word "sorry" if the user has already read the suggested manga. This satisfies the first three criteria for the video.
- The response states, "My program's purpose solves the problem for people who want to read new mangas from certain genres but don't know what to pick."
- The response describes the functionality demonstrated in the video. The response states, "The function of the program is to ask what one of three genres a person wants to read and gives three different mangas from the genre picked, and asks if they have read it or not, and if they want to pick another genre. If the person hasn't read the manga the program will tell them to read it and stop listing mangas of the genre, and then it will ask the user if they want a new genre." Other functionality details are listed.
- The response describes the input and the output. Regarding input, the response states, "The input is when the user puts in what genre they want, and if they have or haven't read the manga, and if they want to pick another genre to choose from." Regarding output, the response states that the program's output "is when the program lists mangas from the genre picked and asks if the user has read them, then apologizes if they have, and asks if the user wants another genre to pick from."

Row 2:
The response earned the point for this row, meeting all three criteria:

- Two distinct code segments are provided, the first showing data being stored in the identified list and the second showing the data being accessed from the identified list.
- The name of the list is identified as `action`.
- The response identifies what is stored in the list. The response states, "the list `'action'` it holds three different action mangas."

Create Performance Task (continued)

Row 3:
The response earned the point for this row, meeting both criteria.

- The response includes code that uses a list of mangas in a certain genre to manage complexity when generating suggested reading. Although the list, `action`, contains only three elements, the for loop can process any number of elements because of the use of this list.
- The response explains how the code would be written differently without the list. The response states, "The program would have to know which variables belong to which genre without being in the list. In the program the code would have to read the variables as `'if jujustuKaisen = action'` and then 'ask have you read this?' and then a `'if read = yes` and `else'` for every single variable that belongs to the genre `action`. It would have to be the same for every other variable for each genre." This response clearly explains how the code would be more complex without the use of the list.

Row 4:
The response did not earn the point for this row, meeting one of the two criteria:

- The response includes a student-developed procedure, `getManga`, with three parameters, `option1`, `option2`, and `option3`. However, none of the parameters are used in the procedure.
- The response describes the functionality of the procedure as it states, "The function iterates through a given list and asks for each item if the manga has been read or not. If the user says they have read the manga then it responds with sorry and continues through the list until it stops. If they haven't read the manga it stops and selects the item not read and tells the user to read the specific item of the list. It loops the lists once the list either stops or the user has not read the manga." The response specifies how the procedure contributes to the overall program, "It contributes to the overall functionality by using all the lists that were set and checks with the user if they have read the mangas in the lists."

Row 5:
The response earned the point for this row, meeting both criteria:

- The student-developed algorithm within procedure `getManga` includes sequencing, selection (multiple if statements), and iteration through use of multiple for loops.
- The response explains, in a minimal amount of detail, how the algorithm in the procedure works so it can be recreated. This description includes each step of the algorithm in order, including getting input from the user ("have the code ask 'What genre? Action, romance, or comedy?' and set the answer to a variable"), the if statement to determine the genre ("If the variable matches one of the genres"), and the for loop inside each if statement ("run a loop that goes through the list and asks if they have read the manga for the genre, and if they have not read the manga tell them to read the item and then stop the block, else tell the user sorry, have this set up for each genre option").

Create Performance Task (continued)

Row 6:
The response did not earn the point for this row, meeting none of the three criteria:

- The response does not describe two calls to the selected procedure that pass different implicit or explicit arguments. Instead, it describes two calls where the procedure obtains two different values for the variable `haveRead` from the user after the call is made. `haveRead` is not an implicit or explicit parameter. An implicit parameter is one that is assigned in anticipation of a procedure call, not after the call is made.
- The response describes the conditions being tested by each call to the procedure as "the manga listed by the function has been read by the user is `true`," and "the manga listed by the function has been read by the user is `false`." However, the conditions do not depend on the value of an implicit or explicit parameter, so this criterion is not met.
- The response identifies the results of each call. For the first call, the response states, "The program will respond with 'sorry' and continue through the list until it reaches the end or until it is `false`." For the second call, the response states, "The program will tell the user to read the item off the list and stops the loop for the list." However, the response to each call does not depend on the value of an implicit or explicit parameter, so this criterion is not met.

Create Performance Task (continued)

Sample Identifier: G

Score:
Row 1: 0
Row 2: 1
Row 3: 0
Row 4: 1
Row 5: 1
Row 6: 1

Row 1:
The response did not earn the point for this row, meeting five of the six criteria:

- The video demonstrates the running of the program showing input (the user's typed responses), program functionality (the program showing the sequences in response to the input), and output (the sequences).
- The response does not describe the overall purpose of the program, but rather describes the program's functionality. The response states, "The purpose is to display custom arithmetic and geometric sequences by setting inputs such as where they start, how long they are, and how much they increment by, or also by displaying the constant Fibonacci sequence." This response is stating what the program does, not its purpose.
- The response describes the functionality demonstrated in the video. The response states that the program "not only listed all of the first eight terms of the sequence through graphic Text objects on the Python 3 Graphics Editor, but also the number of the term through adjacent Text objects," which is functionality shown in the video.
- The response describes the input as "'geometric', '8' terms, initial value: '9', common ratio: '0.8', and the x and y coordination for pressing the Redo button." The response describes the output as "the title, the instructions, the rounded sequence, the term numbers."

Row 2:
The response earned the point for this row, meeting all three criteria:

- The response includes two program code segments. The first segment shows data being stored in the list `fiblist`. The second segment shows the elements in `fiblist` being accessed via a for loop to display them as Text objects on the screen.
- The name of the list is identified as `fiblist`.
- The response describes the data contained in the list. The response states, "A user can display the Fibonacci sequence from 5 to 20 terms." This part of the response makes it clear that the list stores the terms in the Fibonacci sequence.

Create Performance Task (continued)

Row 3:
The response did not earn the point for this row, meeting none of the two criteria:

- The response does not include code that uses a list that helps manage complexity in the code. The values in the list `fiblist` can be simply calculated with a loop without the need for a list.
- The response does not explain how the list manages complexity in the program code. The response states, "Calculating and incrementing a pattern(because each number is the addition of the two previous ones), would have been too complicated to code as two exchanging variables are needed to generate the Fibonacci sequence." This statement is inaccurate because elements in the Fibonacci sequence can be calculated with a loop, without the need to "exchange variables."

Row 4:
The response earned the point for this row, meeting both criteria:

- The response includes a student-developed procedure, `geodisplay`, with three parameters, `initial`, `ratio`, and `n_num`, that are used in the procedure.
- The response describes the functionality of the procedure as it states, "It finds the pattern between each number in the sequence and displays each one(and rounds it to 3 decimal places if needed) as Text on the canvas individually through a loop." The response describes how the identified procedure contributes to the overall functionality. The response states, "The function gives a user a wider range of options to learn in the program in addition to arithmetic sequences."

Row 5:
The response earned the point for this row, meeting both criteria:

- The student-developed algorithm within procedure `geodisplay` includes sequencing, selection (if statement), and iteration (for loop).
- The response explains in great detail how the algorithm in the procedure works so it can be recreated. For example, the response states, "`Geodisplay` function takes three numerical parameters meaning the initial value, the number of terms, and the common ratio of the sequence. First, you set a variable called `number` and set it equal to the initial value. Then you start a for loop that increments another variable `i` by `1` for each run of the loop, from `0` to the number of terms. For each iteration you display a different Text object that will display `number` rounded to 3 decimal places if needed." More detail is given for the other steps of the algorithm.

Create Performance Task (continued)

Row 6:
The response earned the point for this row, meeting all three criteria:

- The response describes two calls to the procedure with different arguments. The response states, "In `geodisplay(10, 2, 9)`, the user could instruct a geometric sequence with an initial value of 10, a common ratio of 2, and a sequence length of 9 to display." The response describes the second call as, "In `geodisplay(9, 0.9, 5)`, the user could instruct a geometric sequence with an initial value of 9, a common ratio of 0.9, and a sequence length of 5 to display." The response describes two calls, one that will cause the body of the if statement to execute, and the other that will not.
- The response describes the conditions being tested for the two procedure calls. The response states the first call is "`Geodisplay(10, 2, 9)` has a common ratio(the middle parameter) of `2`, which is greater than `-1` but also NOT less than `1`, so the condition is not satisfied." The response also states the second call is "`Geodisplay(10, 2, 9)` has a common ratio(the middle parameter) of `0.9`. The code predicts that the sequence eventually converges to 0 because the ratio is less than `1` and greater than `-1` (`0.9`), so it also displays the text 'eventually converges to zero'."
- The response identifies the results of each call. The response states that the first call results in "The numbers displayed are `10`(the initial value), `20`, `40`, `80`, `160`, `320`, `640`, `1280`, and finally `2560`(9 terms)." The response states that the second call results in "Specifically, the numbers displayed are `9`, `8.1`, `7.29`, `6.561`, and finally `5.905`(5 terms)."

Create Performance Task (continued)

Sample Identifier: H

Score:
Row 1: 1
Row 2: 1
Row 3: 0
Row 4: 0
Row 5: 0
Row 6: 1

Row 1:
The response earned the point for this row, meeting all six criteria:

- The video demonstrates the running of the program showing input (mouse clicks on the program buttons), program functionality (the user navigating the adventure game), and output (messages to the user, different backgrounds).
- The response describes the program's purpose as "provid[ing] the user with the entertaining experience of a chose-your-adventure style game mixed with an RPG-like stat system."
- The response describes the functionality demonstrated in the video by describing how the game is being played. The response states, "The program allows the user to click with the mouse on specific buttons with specific texts which can change the screen, change values of variables, and change images URL, leading to the next step of the story."
- The response describes the input and the output, though it does not explicitly label them as input and output. Regarding input, the response states, "The program allows the user to click with the mouse on specific buttons." Because the response describes the user as doing the clicking, it is clear that this is input. Regarding output, the response states that the user's clicks will "change the screen, change values of variables, and change images URL." Because these changes are described as being in response to the input (which the "program allows"), it is clear that this is the program's output.

Row 2:
The response earned the point for this row, meeting all three criteria:

- The response includes two program code segments. The first shows how data is stored in the list `stats`. The second shows the list `stats` being used as part of the program's purpose. The values of all three elements in the list are being shown in the console to show various statistics about the character.
- The name of the list is identified as `stats`.
- The response describes the data contained in the list named. The response states, "It represents the Strength, intelligence, and Harmony of the user's character."

Create Performance Task (continued)

Row 3:
The response did not earn the point for this row, meeting none of the two criteria:

- The response includes a program segment that uses a list. However, the included program segments do not show how this collection type manages complexity because it would be just as simple to use three variables for the three values in the list.
- The response does not correctly describe how the list `stats` manages complexity. The response states that if the code used three variables instead of a list "it would lead to the code being way harder to write due to the larger amount of variables and it would also lead to the code becoming way harder to understand." However, this statement is not correct. Only three variables would be required, and the code would likely be easier to understand if named variables were used for the Strength, Intelligence and Harmony levels, instead of accessing these values from a list named `stats`.

Row 4:
The response did not earn the point for this row, meeting one of the two criteria:

- The response includes a student-developed procedure, `dragon`, and shows this procedure being called. However, this procedure does not contain an explicit parameter. The variable `dragonhealth` is an implicit parameter.
- The response describes what the identified procedure does. The response states that it "checks the `dragon_health` value and runs a line of code depending on said value." The response also states how this procedure contributes to the overall functionality of the program. The response states that the procedure runs "in order to help the user understand that his/hers choosen stat is high enough to succesfully attack the dragon trough a visual and audio queue."

Row 5:
The response did not earn the point for this row, meeting one of the two criteria:

- The student-developed algorithm within the procedure `dragon` includes sequencing and selection (if-else statements) but does not contain iteration.
- The response describes the algorithm in enough detail that someone else could recreate it. The response includes a description of each step of the algorithm starting with the description, "The algorythm,trough an if statement, checks if the value of a predetermined variable called `dragon_health` is equal to three and if the value is found to be equal to three nothing happens." The response then describes the next step as, "then the algorythm moves to the first else if statement which checks if the `dargon_health` value is equal to `2` and if the value is found to be `2` the algorythm will set the image URL for `'checkmark1'` to 'checkmark.jpg' and trigger a sound queue." This description continues for the remaining values of `dragon_health` (`1` and `0`) as shown in the procedure.

Create Performance Task (continued)

Row 6:
The response earned the point for this row, meeting all three criteria:

- The response describes two calls to the selected procedure that pass different arguments via the implicit parameter `dragon_health`. For the first call, the response states that it "checks to see if the `dragon_health` value has been lowered to two" meaning it is calling the procedure when `dragon_health` is `2`. For the second call, the response states that it "checks to see if the `dragon_health` value has been lowered to zero," meaning it is calling the procedure when `dragon_health` is `0`. These two values of `dragon_health` cause different if statements to execute in the procedure `dragon`.
- The response describes the conditions being tested by the procedure in each call. The response gives the first call's condition as "the call checks if the value of dragon health has been decreased to `2`." The procedure is expected to execute the code that corresponds to `dragon_health` having a value of `2`. Similarly, the response gives the second call's condition as "the call checks if the value of dragon health has been decreased to `0`." In this call the procedure is expected to execute the code that corresponds to `dragon_health` having a value of `0`.
- The response describes the result of each call. The response states that the result of the first call is to "set the image URL for `'checkmark1'` to 'checkmark.jpg' and trigger a sound queue," while the result of the second call is to "set the image URL for `'checkmark3'` to 'checkmark.jpg', trigger a sound queue, stop battle music, and change screen to the victory screen."

Create Performance Task (continued)

Sample Identifier: I

Score:
Row 1: 1
Row 2: 0
Row 3: 0
Row 4: 1
Row 5: 0
Row 6: 0

Row 1:
The response earned the point for this row, meeting all six criteria:

- The video demonstrates the running of the program, showing input into the program through clicking a button, and pressing the space bar to make the character jump. Output includes the character moving on the screen. This satisfies the first three criteria.
- The response specifies the program's purpose is to "train and increase hand to eye coordination."
- The response describes the functionality demonstrated in the video. The response states the functionality as, "to jump when an obstacle is head towards the player. This is done when every time you press the spacebar you glide up wards at a set speed. As demonstrated in the video if the player is unsuccessful the players character will die, and the game will end."
- The response describes the input and the output. Regarding input, the response states, "This is done when every time you press the spacebar you glide up wards at a set speed." Regarding output, the response states, "When the player comes in contact with an obstacle the player dies and explodes which can be seen at 0:00:11, ending the game. When the player reaches the score 10, then the cow comes to the player and displays a congratulations text that acknowledges the win."

Row 2:
The response did not earn the point for this row, meeting one of the three criteria:

- Two distinct code segments are provided. The first segment shows use of data in a list named `list`. However, in the second code segment, the same code is shown. The code segments do not show data being stored in a list.
- The name of the list is not identified.
- The response correctly states, "The data in my list is named `'1'` and `'2'`," identifying what is stored in the list.

Create Performance Task (continued)

Row 3:
The response did not earn the point for this row, meeting none of the two criteria:

- The response includes a program code segment that uses a list, but the list is not used to manage complexity. The use of the list is merely to act as a random integer between 1 and 2. The list is irrelevant and can be replaced with a simple Boolean variable.
- The response does not explain how the list manages complexity by showing how the code would be more complex without the list. The response states, "Another way of allowing for the change with costumes would have a Boolean expression so when the player gets over the obstacle and gain another point to their score the costume will switch." However, this does not reduce the complexity in any substantial manner since the list only has two elements.

Row 4:
The response earned the point for this row, meeting both criteria:

- The response includes a student-developed procedure, `game; time`, with one explicit parameter, `time`, and a call to this procedure in a second code segment using the argument `2.66`.
- The response describes the functionality of the procedure. The response states the functionality as "it also controls when the game stops, if the score reaches ten then the game stops and the cowboy wins, and the cow will glide to the cowboy. The score increase based on the obstacles you jump over; you have to be able to jump over ten of them to win." It also describes how it contributes to the overall program. The response states, "The procedure created was used to accurately control the speed of the game as well as other aspects of the game."

Row 5:
The response did not earn the point for this row, meeting one of the two criteria:

- The student-developed algorithm within procedure `game; time` includes sequencing, selection (an if statement), and iteration (a for loop). However, the selection is trivial. The for loop will always run exactly 10 times (unless the code is halted by an external event), so if the end of the for loop is reached, the value of `score` will always be `10`. Thus, the if statement is not needed.
- The response explains how the algorithm works in five detailed steps. It states that "Step 1, the procedure controls the obstacles used in the game; they are differed from one to ten. Step 2, switch costume randomly through the use of the list. In the list is the two costumes, the vulture and the cactus. Step 3, the obstacles are then set to go to x:250 y: -115, then glide to x: -300 y: -155 at the speed of the set time in the parameter." Steps four and five are also described.

Create Performance Task (continued)

Row 6:
The response did not earn the point for this row, meeting none of the three criteria:

- The response gives two different values for the time parameter, but this does not cause two different segments of the code to execute.
- The response describes the conditions being tested for the cases, but these reference the overall score and not the parameter for time, and the conditions being tested are the same. The response states that the conditions being tested are “if the score received is equal to ten” and states these conditions as “the score received is equal to ten” and the “second call to procedure are the same as the first.”
- The response describes that the results for both calls will result in the same outcome. The response states, “The result to the first call to procedure the procedure is if the score is equal to ten then the procedure will broadcast win, which is a congratulations pop up that shows proof of the win.” The response describes the result of the second call as, “The result of the second call to procedure is the same as the first call to procedure, if the score is equal to ten then procedure broadcast win.” This does not satisfy the criteria since the parameter does not lead to different code being executed, leading to different results.

Create Performance Task (continued)

Sample Identifier: J

Score:
Row 1: 0
Row 2: 1
Row 3: 0
Row 4: 0
Row 5: 0
Row 6: 0

Row 1:
The response did not earn the point for this row, meeting four of the six criteria:

- The video demonstrates the running of the program showing input (text input), program functionality (translation from text to binary representation), and output (messages to the user, binary representations).
- The response does not describe the overall purpose. The response states, "The intended purpose of this program is to tell the user what any lowercase letter or full word would be in binary." However, this statement is phrased such that it describes the function of the program, not its purpose. A purpose could have been "to help the user learn binary representations of letters" or "to translate between letters and binary," but instead the response lists what the program does.
- The response describes the functionality demonstrated in the video. The response states, "The program asks a question to get input from the user and then outputs the input in a binary format."
- The response describes the program's output demonstrated in the video as "the input in a binary format." However, the response does not clearly identify the input. The response states that the program "get[s] input from the user" and that the user "input[s] specific data," but it does not specify what input is demonstrated in the video.

Row 2:
The response earned the point for this row, meeting all three of the criteria:

- The response includes two program code segments. The first shows data being stored in the list `a2m` when the variable is declared. The second shows each of the elements in `a2m` possibly being accessed in the procedure `word2binary`, depending on the value of `x`, which is input from the user.
- The name of the list is identified as `a2m`. The response also identifies another list, `n2z`, which is not used to score this row since two lists were identified. Scoring is based on the first list identified.
- The response describes the data contained in the list `a2m`. The response states, "the data contained in the list `a2m` is all lowercase letters in binary from `a` to `m` saved in a string."

Create Performance Task (continued)

Row 3:
The response did not earn the point for this row, meeting none of the two criteria:

- The response does not include code that uses a list that helps manage complexity. The list is being used to store the binary values of the letters a through m. However, the list does not help manage complexity in the code shown. Because of the way the code is written, it would have been just as simple to print the literal binary string from the body of each if statement. For example, `if (x == "a") cout << "01100001"`.
- The response does not explain how the list manages complexity in the program code. The response states, "If the code were to be written with no list then the code would be much longer and more confusing due to not knowing which letter equals which number on the list." This statement is not accurate. As described above, the code would be the same length, and no more confusing (in fact, perhaps less confusing) if the binary strings were printed directly from the body of the if statements.

Row 4:
The response did not earn the point for this row, meeting one of the two criteria:

- The response includes a student-developed procedure, `word2binary`, but this procedure does not take an explicit parameter. The variable `x` used in the procedure is an implicit parameter.
- The response describes what the identified procedure does. The response states "The procedure takes the input of the question and checks to see if it's equal to `'=='` any of the string values. If it is equal to a stored string value the procedure outputs the input string into binary using the list." The response also states how this procedure contributes to the overall functionality of the program. The response states, "the general procedure of the program consists of reading users' input of their 'favorite lowercase letter' and printing it out in binary using the list element." This part of the response makes it clear where the `word2binary` procedure fits into the overall functionality of the program.

Row 5:
The response did not earn the point for this row, meeting one of the two criteria:

- The student-developed algorithm within the procedure `render` includes sequencing and selection (if-else statement) but does not include iteration.
- The response describes the algorithm in enough detail that someone else could recreate it. The response describes the algorithm as "taking the value of the user's input and outputting the corresponding array value if the comparison between the saved string and input is `true`, the code will print out a list element." The response gives an example that would be followed for all letters, "For example, if the input `'a'` is equal to the string value `'a'` then output `[0]` from array `a2m` as `0` is the first value of the array and a is the first letter in the alphabet. It repeats this for all possible letters in the alphabet and only stops when `a` is `==` to the inputted letter by using if and else if statements to check every value until there is a match."

Create Performance Task (continued)

Row 6:
The response did not earn the point for this row, meeting none of the three criteria:

- The response does not describe two separate calls to the selected procedure. Instead, it describes two cases as the first example, and what happens in the calling function as the second example. In describing the first call, the response states, "for example, is the user typed lowercase letter `a` then the function prints `0110001` but if the user were to type `b` the function would output `01100010`," listing both options as part of the first call to the procedure. The response describes the second call as "The function gets called again in a for loop and this time reads different input data which causes different outputs to occur." The response is describing the operation of the given procedure as one test case and the operation of the calling procedure as the other test case, which is incorrect.
- The response does not clearly describe the conditions being tested by the procedure. The response lists the first call's condition as "the input variable equal to `'=='` the string" and the second call's condition as "'is the letter a lowercase letter' and if so which letter." These statements are two different ways of describing the same condition, and because there are not two clearly defined calls, it is not clear how the condition is different between the two calls.
- The response does not describe the result of each call. The response states, "There are a total of 27 possible outcomes that can come from the first call of the function 26 of them come from the input actually being `true` to at least one string. The one remaining output is for when the input is not equal to `'=='` anything, leaving the procedure to output space." However, it does not give the actual expected output for any given input.