2021

AP® CollegeBoard

AP® Computer Science Principles

Sample Student Responses and Scoring Commentary

Inside:

Performance Task—Create

☑ Scoring Guideline

☑ Scoring Commentary

Student Samples provided separately

Create Performance Task — 6 points

Learning Objectives: **CRD-2.B** **AAP-1.D.a** **AAP-1.D.b** **AAP-3.C** **AAP-2.H.a** **AAP-2.K.a** **CRD-2.J**

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)** **4.A**	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).
Row 2 **Data Abstraction** **(0-1 points)** **3.B**	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)** **3.C**	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)** **3.B**	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, use the first procedure 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)** **2.B**	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)** **4.C**	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 segment of code in the algorithm to execute even if the result is different. • 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 and is 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 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. The student should be able to demonstrate the program running and explain its purpose, how it functions and how it handles input and output of information.

Programs typically process collections of data of the same type 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 to hold data that is critical to fulfilling the program's purpose. Students must explain why the list manages complexity in the program 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 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).

Sample: 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 program receiving both players' column integer as input and displaying the 'X' or the 'O' in the appropriate column if the column exists as output. The functionality demonstrated is the continuous playing of a connect-four game. This satisfies the first three criteria for the video.
- The program's purpose is to "facilitate a 'connect-four' game play experience between two players."
- The response describes the functionality demonstrated in the video as "the program prompts Player X to type the column they'd like to play in, and re-prompts the player until they enter an accepted value."
- The response describes the input demonstrated in the video as "inputs are given by Player X and Player O, corresponding to the desired column they'd like to place their piece," and the output is "If a player achieves four-in-a-row, the program additionally outputs a statement identifying the winner then terminates."

Create Performance Task (continued)

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

- Two distinct code segments are provided: one showing storage of data in a list named `grid`; and a second one showing how `grid` "is accessed within a function as an argument and given the local name '`current_grid`.'"
- The name of the list is identified as `grid`. The response further explains how `grid` is passed to a function that has an argument named `current_grid`. This makes clear the connection between the two code segments, so it is clear that there is only one list that is being used to respond to this question.
- The response states that the data "represents the 'status' of each position on the connect-four game board." The response also states, "The 'status' is denoted by an 'X' (piece belongs to player X), an 'O' (piece belongs to player O), or a ' ' (position is empty)."

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

- The response includes a program code segment that shows the `grid` list being used to manage complexity in the program because the list access, and indexes, enable the current element of the list to be checked easily.
- The response explains why the program could not be written without the list. The response states that "without the list, the program would have no way to remember previous player inputs, making gameplay impossible." This response is plausible given the program described.

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

- The response includes a student-developed procedure, `check_overall`, that contains two parameters, `current_grid[6][7]` and `player`. Both parameters are used in the procedure. Additionally, the response includes an example call to the procedure `check_overall` that passes the arguments "`grid`" and "`current_turn`" to the parameters.
- The response describes what the procedure `check_overall` does by stating that it "will access the item `current_grid[i][j]` and will determine whether a vertical, horizontal, or diagonal sequence of four pieces of the same owner stems from that position." This check is done by a helper procedure `check_individual`. The response states that the procedure "contributes to the overall functionality of the program by checking after every player's turn to see if they've won."

Create Performance Task (continued)

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

- The student-developed algorithm within procedure `check_overall` includes sequencing, selection (nested `if` statements), and iteration (nested `for` loops).
- The response explains how the algorithm works. It states that it "works by using a '`for` loop' that runs six times, then nesting another '`for` loop' inside that runs seven times. The outer loop will initialize the variable '`i`' as 0 and increment by +1 each time it runs, while the inner loop will initialize the variable '`j`' as 0 and increment by +1 each time it runs." The response goes on to describe that selection is used "If the piece belongs to '`player`', `check_overall` will call the `check_individual` function and pass in '`i`,' '`j`,' and '`current_grid`' as arguments."

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

- The response describes two calls to the procedure. In the first call, `grid` "has been updated so all positions have the status " " except for one position which has status '`X`'." This `grid` and '`X`' are passed as arguments to the procedure. In the second call, a new `grid` "has four '`O`' pieces in an uninterrupted vertical line." This `grid` and '`O`' are passed as arguments to the procedure.
- The response describes the conditions as being "whether four positions in a row of any kind (horizontal, etc.) are occupied by '`X`' when there is only one '`X`' in the `grid` and "whether four positions in a row of any kind are occupied by '`O`'" when there is four '`O`' in a row in the `grid`.
- The response describes the result given when the procedure is tested with a `grid` with only one '`X`' and the value '`X`' as the "the function returns 2", with the main calling function "proceeding to the next player's turn." The response describes the result given when tested with a `grid` with four '`O`' in a vertical line as "the function identifies the existence of four-in-a-row for Player O and returns 1," with the main calling function "breaking the loop and ending the game."

Create Performance Task (continued)

Sample: 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 program receiving the player's mouse clicks as input and displaying the order of the targets hit at the game's end, as well as the number of "`targetsHit`" and a score as output. The functionality demonstrated in the video is the user attempting to click on targets and then a recap of the targets that were selected. This satisfies the first three criteria for the video.
- The response describes the program's purpose as "training a user's mouse reflex skills if coming from the older generation or simply someone who may have a slow time clicking on links."
- The response describes the functionality demonstrated in the video as "the clicking mechanic based on a mouse cursor clicking on a target to earn points."
- The response describes the input demonstrated in the video is "the mouse clicks are inputs that will work only if clicked on a sprite", and the output is "the user will be granted his or her history, in chronological order, of all types of targets that were clicked."

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

- Two distinct code segments are provided: one showing storage of data in a list named `historyT`; and a second one showing how `historyT` is accessed within a code segment which "corresponds to one of the targets with a value of +1 when clicked."
- The name of the list is identified as `historyT`.
- The response states that the data "represents the specific targets that were hit in chronological order."

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

- The response includes a program code segment that shows the `historyT` list being used to manage complexity in the program by keeping track of the order that targets are being hit.
- The response explains how the list `historyT` manages complexity in the program. It states, "because at the very end, the user can see each target that shows up on the screen to know what targets were hit throughout the ten seconds." The response also states, "Without this list, it would be difficult because the variable `block` would only track the amount but not order and say and ask/answer blocks would require some extra code and may get overwhelmed by the large inputs of multiple target values."

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

Create Performance Task (continued)

- The response includes a student-developed procedure, `diffMod`, that contains a parameter, `difficulty`. The parameter is used in the procedure. Additionally, the response includes an example call to the procedure `diffMod` that passes the argument "`answer`" to the parameter.
- The response describes what `diffMod` does by stating that it "accounts for the difficulty modifier that is accomplished by a parameter that accepts values based on a ask/answer block where one target can change in size or the wait time with specific boundaries to prevent a sprite to be at the edge" The response states that the procedure "contributes to the overall functionality of the program because without the program the `output:score` and `output:targetsHit` cannot be updated because the two inputs `mouse click` and `difficulty` cannot be established."

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

- The student-developed algorithm within procedure `diffMod` includes, sequencing, selection (`if/else` statement), and iteration (`repeat until`).
- The response explains how the algorithm works. It states that "The procedure's parameter accepts values greater or less than 1 when called based on an ask/answer block. The procedure demonstrates a selection where a value greater than 1 enables the harder difficulty. Both outcomes are similar with some numbers adjusted." The response goes on to describe that "There is a timer paired with a `repeat until` loop and an operator block for the game to continue running for a set amount of time."

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

- The response describes two calls to the procedure, `diffMod`: one for hard difficulty; and one with easy difficulty.
- The response describes the conditions as being "If the user inputs the number 2, then the procedure will run the '`then`' portion of the command and not the '`else`' portion because the value is greater than 1. Hard Difficulty is enabled." and "If the user inputs the number 0, then the procedure will run the '`else`' portion of the command and not the '`then`' portion because the number is less than 1. Easy Difficulty is enabled."
- The response describes the result being tested for the hard difficulty as "a smaller size for targets, a greater boundary to spawn since the sprites are smaller, and a smaller initial wait time for how long a target will stay on screen.", and describes the result being tested for the easy difficulty as "a larger size for targets, a smaller boundary to spawn since the sprites are bigger, and a bigger initial wait time for how long a target will stay on screen."

Create Performance Task (continued)

Sample: C

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 program receiving the player's text entries and key presses as input and displaying the ball's movement, a message when the ball hits a wall, as well as the results of the first run (points earned) as output. The function is demonstrated as a ball being moved through a maze; when it touches certain objects, it earns points, and when it touches the wall, it goes back to the beginning. This satisfies the first three criteria for the video.
- The response describes the program's purpose as "to entertain the user through a fun obstacle course in which the user has to move a red ball from the starting point to a golden mushroom without bumping into the walls, and while also earning points."
- The response describes the functionality demonstrated in the video as "each time the user's red ball bumped into a wall the ball was back at the starting position and the game restarted." The response also states, "The video also shows the end results page for the first run of the game in which the player earned 200 points, and took two tries to finish."
- The response describes the input demonstrated in the video as "the mouse keys which move the red ball throughout the screen, and the mouse clicks used to proceed from the welcome screen," and the output is "a result screen with the number of points the user earned, and the tries that they took to finish."

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

- Two distinct code segments are provided: one showing storage of data in a list named `score_point_list`; a second one showing how `score_point_list` is processed within an `if` statement.
- The name of the list is identified as `score_point_list`.
- The response states that the data is "the coordinates of all the stars and mystery blocks which allows the program to identify where these items that give the user points are located."

Create Performance Task (continued)

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

- The response includes a program code segment that shows the `score_point_list` being used to manage complexity in the program by storing the location of the objects that give the user points and using a `for` loop to determine if the user hits any of the objects while going through the maze.
- The response explains how the list `score_point_list` manages complexity in the program. It states, "it allows it to check if the user moved the red ball onto the stars or the mystery blocks all at once in a much quicker manner, through the use of a `for` loop and an `if` statement." In addition, the response explains how the program code would need to be written differently if the list wasn't used by stating, "If not for the list, I would've had to create several separate `if` statements for each of the different coordinate points stored in the list, and check if that equals the current location of the red ball."

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

- The response includes a student-developed procedure, `move_character` that contains three parameters; `event`, `name`, and `speed`. The parameters are used in the procedure. Additionally, the response includes an example call to the procedure `move_character` that passes the arguments "`event`," "`"Player: " + player`," and "`5`" to the parameters.
- The response describes the purpose of `move_character` by stating that it "moves the red ball depending on which mouse key the user inputted through the use of selection." The response states that the procedure contributes to the overall functionality of the program because "If the user pushes the right key the ball moves to the right a certain amount, if user pushes the left key it moves to the left, if user enters the up key the ball moves up, and if user pushes the down key it moves down."

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

- The student-developed algorithm within procedure `move_character` includes sequencing, selection (`if/elif` statement), and iteration (`for` statements).
- The response explains how the algorithm works. It states, "After each time the ball moves, the procedure uses a `for` loop and `if` statements to check if the ball is on a coordinate of the walls, or the score items such as the stars, by accessing the lists that these coordinates are located in and by checking if even one of those coordinates matches the balls current coordinates." The response goes on to describe that "[i]f the coordinates of the ball matches a coordinate of the wall, the procedure lets the user know that they bumped into a wall and makes them restart with zero points, however if the ball's coordinates match one of the coordinates of the score items, then it lets the user know that they got some points. It keeps track of these points by appending a number to a list each time the user gets a point."

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, `move_character`: one with a `speed` parameter set to 5; and one with a `speed` parameter set to 15.
- The response describes the conditions as "checking if the `speed` parameter is equal to 5 or if it equals 15 will execute the first part of this selection statement" and checking "if the `speed` parameter is equal to 5 or if it equals 15, will execute the second part of this selection statement".
- The response describes the results being tested as displaying "the difficulty of the game is medium at this level, and with this argument the red ball moves in increments of 5", and "that the difficulty of the game is hard at this level, and with this argument the red ball moves in increments of 15."

Create Performance Task (continued)

Sample: 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 input by the program user selecting their age using up and down arrows, selecting a movie genre, and then as output, the program displays a list of movies they should watch based on age and genre. The functionality being demonstrated is showing a list of movies appropriate for different age groups and genre preferences. This satisfies the first three criteria for the video.
- The response describes the program's purpose is as suggesting for users "either 3 action or 3 comedy movies that are age-appropriate."
- The response describes the functionality demonstrated in the video is "When I inputted my age and selected a genre, I was suggested 3 different movies accordingly. When I restarted the program and increased my age, I received different, more mature movie suggestions that are appropriate for the given age. Every time I restarted the program, it eliminated the previously displayed movie suggestions to make room for the new ones in the movie list. Additionally, a happy noise is exerted once the movie suggestions are given."
- The response describes the input and output demonstrated in the video are described as, "I click on the up or down arrows for the age input, the output number increases or decreases by 1 accordingly. When I click which genre I want: comedy or action, the screen switches to the movie output screen which from there randomly displays movies according to my given age and genre choice."

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

- Two distinct code segments are provided: one code segment shows data being stored in the list; the second code segment uses the movie lists to process output from the movie list based on the age entered by the user to fulfill the program's purpose. Note there are several lists presented; however, the response identifies a specific list to focus on, and this list is used in the second code segment.
- The response identifies multiple lists, so the first list identified in the written response is used. The response identifies the name of the list as `youthAction`.
- The response identifies and describes what is contained in `youthAction` as "my lists store the names of action movies." The list fulfills the program's purpose when "these movies are randomly selected and suggested to the user (when the user inputs age and genre choice)."

Create Performance Task (continued)

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

- The code segment demonstrates the list being used to manage complexity by generating a list of movies based on the user's input age and genre, which can then be used to randomly select three movies to suggest.
- However, the written response generically describes how lists can be used in programs to manage complexity. The response does not explain specifically how the selected list manages complexity for this particular program by either explaining why the program code could not be written or how it would be written differently without using a list.

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

- The response includes a student-developed procedure `getMovie` that contains a parameter `genre`. The parameter `genre` is used in the procedure. Additionally, the response includes an example call to the procedure `getMovie` that passes the argument "`Comedy`" to the parameter.
- The response describes what `getMovie` does and how it contributes to the overall functionality of the program: "the code is using the input of the user's age and genre choice to from there, randomly select what three movies the user will be suggested. Without this function in the program, the user's genre and age would not be implemented into the overall decision of what list to pull movies from."

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

- The student-developed algorithm within procedure `getMovie` includes sequencing, selection (`if/else`), and iteration (`for`).
- The response explains how the algorithm works with enough detail that someone else could recreate it. It states that "Using an `if` statement, it looks at the user's age. If the user is less than or equal to 12 years old, regardless of their genre, they will be suggested movies from either of the youth lists. If the user is NOT younger than 12 and they are between the ages of 13 and 19, they will be suggested movies from either of the teen lists. And if they are still not in either of those age categories, their movies will be pulled from either of the adult lists." The response goes on to describe how the user's age is processed using a selection statement, and iteration is used "randomly select and append three different movies from that appropriate age and genre list to the empty list movie."

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

- The response describes two calls to the procedure `genreChoiceScreen`: one call with the argument of "`Comedy`"; and one with the "`Action.`"
- The response describes the conditions as being whether the user has selected Comedy or Action.
- The response states that the result of the first call "suggested three movies from one of the three different Comedy lists," and that the result of the second call "suggested three movies from one of the three different Action lists."

Create Performance Task (continued)

Sample: E

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

Row 1:
The response did not earn the point for this row. The response does not meet one of the six criteria:

- The video demonstrates input by the user clicking the start button and the timer starting. The player initially lost because they clicked forward when the background turned red (an example of output), but their second attempt demonstrated the player winning with the winning message output at the end. The functionality that is shown is the user playing a game to only move the ball when the screen is green and not when it is red. This satisfies the first three criteria for the video.
- The response does not describe the program's purpose by providing either the problem being solved, or the creative interest being pursued through the program. Rather, the written response's purpose described part of the functionality.
- The response describes the functionality demonstrated in the video as: "When you push the button the ball moves, if you push it when the screen is red it sends it back to the bottom."
- The response describes the input as the "finger touching the forward button" and the output as "the time it gives you, and the ball moving."

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

- Two distinct code segments are provided: one code segment shows data being stored in the list `Good_things_list`; the second code segment uses the same list to pick a random element that fulfills the program's purpose of displaying a good message if the player wins.
- The name of the list is identified clearly as `Good_things_list`.
- The response correctly describes the data in the list as "the good word that is said after your time is given to you in the app."

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

- The response includes a program code segment that shows a list being used to manage complexity by storing a set of winning messages.
- However, the response does not explain how the list manages complexity and states that "this section of my program could not be written because it requires the selection from the list to get different good words to say." It is possible to write this section of code without a list. One way to accomplish this would be to use a sequence of `if` statements to display a single message depending on the value of a random number generated.

Create Performance Task (continued)

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

- The response shows a student-developed procedure, `Color2`, with an explicit parameter, `x`, which is used in the body of the procedure to determine the outcome of the `if` statement. The response also contains a code segment where the procedure `Color2` is called twice in the `if/else` block.
- The response states what the procedure does to contribute to the overall functionality of the program by stating that the " identified procedure decides whether the ball moves forward or if it gets sent back to the beginning based on what color the screen is."

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

- The program code segment representing the student-defined algorithm (procedure `Color2`) includes sequencing and selection, but it does not include iteration. The code does not use a loop of any kind.
- The response does include steps on how the algorithm works in enough detail to recreate the algorithm: "If the screen is green then it will call ball 1 to move by any number 15-20, if the screen is red then it will set the balls Y value to 500."

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 that cause different segments of code to execute. The first call is to `Color2` with a parameter of `green` which causes the `then` part of the `if` statement to execute. The second call is to `Color2` with a parameter of `red` which causes the `else` part of the `if` statement to execute.
- When asked to describe the condition(s) being tested by each call, the response states, "the condition tested in the call is the color of the screen." This part of the response doesn't clearly describe the condition; however, when asked about the result of the call, it states, "if it is green" for the first call and "if it is red" for the second call.
- The response identifies the result of each call. For the first call, it states, "If it is green the ball moves forward." For the second call, it states, "If it is red then the ball would be sent back to the bottom of the screen."

Create Performance Task (continued)

Sample: F

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

Row 1:
The response did not earn the point for this row. The response only met five of the six criteria:

- The video demonstrates the program running and the user entering the number of hours they have listened to music each day (input). Total hours input are calculated (program functionality) in order to printed messages in two textboxes (output). This satisfies the first three criteria for the video.
- The response does not describe the program's purpose by providing either the problem being solved or the creative interest being pursued through the program. The response states the program's purpose is to "tell the user how many hours of music they've listened to in a week based on the numbers they've inputted." However, this is the functionality of the program.
- The response describes the functionality demonstrated in the video as "user can input a certain number of hours and click the add button so the number will be added to the total number of hours they have listened to that week."
- The response describes the input and output demonstrated in the video as, "inputs of the video are each time the user types and adds a number and the output is the number the code calculates and puts on the screen as well as the message that comes up only when the total number of hours is greater than 27."

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

- Two distinct code segments are provided: one appends input values to `hours` list; the second one shows the use of `hours` to track total hours of music for the week to fulfill the program's purpose.
- The name of the list is identified as `hours`.
- The response states that the data in the "`hours` list represents each hour amount the user inputs by using the add button."

Row 3:
The response did not earn the point for this row. The response does not meet either of the two criteria:

- The use of the list `hours` does not manage complexity in the program, as the individual values are being stored only to compute a sum, which could have been done with one variable computing a running total.
- The response states, "the hour list manages complexity because it allows any number of inputted hours to be stored in the list instead of using individual variables for each time the user wants to add to the lists." However, the code could be written with a single variable that is used to keep track of a running total of hours instead.

Create Performance Task (continued)

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

- The response includes a student-developed procedure `findTotal` that contains a parameter `hours`. The parameter `hours` is used in the procedure. Additionally, the response includes an example call to the procedure `findTotal` that passes the argument "`hours`" to the parameter.
- The response describes what the function does in the program by stating it calculates "the total number of hours of music the user has listened to" and tells the user when "they've listened to more music than the average person per week if their total is greater than 27."

Row 5:
The response did not earn the point for this row. The response met only one of the two criteria:

- The response includes a program code segment of a student-developed algorithm found in the body of the `findTotal` procedure. This algorithm includes sequencing, selection (`if`), and iteration (`for`).
- The response does not accurately explain how the algorithm sequence works when the procedure is called. The part of the explanation that states, "if the inputted hour number is equal to the hours at the specific index it becomes the new total," does not match the included program code.

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

- The response describes two calls to the procedure `findTotal`: one with the user passing the values "5,1,7,3,2,1 and 4"; and one with the user passing the values "6,4,9,5,3,2, and 5."
- The response describes the conditions as being whether the user has selected entered values that total a value less than 27 hours, and the second user entered values greater than 27.
- The response identified the results of the first call being tested by stating, "the user will see 'You've listened to 23 hours of music this week' in the first text area." For the second call being tested the response identifies the result as "the user will see 'You've listened to 29 hours of music this week' in the first text area and 'Wow! You've already listened to more music than the average person per week!' in the second text area."

Create Performance Task (continued)

Sample: G

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

Row 1:
The response did not earn the point for this row. The response met only five of the six criteria:

- The video demonstrates input by the user selecting the type of quote for either movie line or song lyric, which produces as output a list of associated favorite quotes. The functionality in the video is the quotes appearing when the different buttons are selected. This satisfies the first three criteria for the video.
- The response does not describe the program's purpose by providing either the problem being solved or the creative interest being pursued through the program. The response states the purpose is "to display some of my favorite song lyrics and movie lines for the user." However, this is a description of the overall function of the program.
- The functionality demonstrated in the video is "the user clicking on each of the buttons and the correct information being output."
- The input and output of the program demonstrated in the video, showed input as the "user clicking one of the buttons" and output as "a list of either song lyrics or movie lines, is displayed as text."

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

- Two distinct code segments are provided: one in which `quoteType` shows how the data is stored; the second one shows the use of `quoteType` to output the quotes to `outputText` to fulfill the program's purpose.
- The name of the list is identified as `quoteType`.
- The response states that the data "stores the type of each of my quotes (either song lyric or movie line)."

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

- The response includes a program code segment that shows a list, `quoteType`, being used to manage complexity in the program by storing an arbitrary number of quotes and using a loop that repeats based on the number of elements in the list (using `quoteList.length`).
- However, the written response does not explain why the code cannot be written without a list, nor how it would be written differently without using the list. The response states: "Without my list differentiating the two options, all of the quotes would be printed out together when either of the buttons are pressed." The response does not explain why they cannot filter on a type any other way without the list or write the code in a more complex manner to achieve the same results. It merely says how the program will run incorrectly without the list.

Create Performance Task (continued)

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

- The response includes a student-developed procedure `filter`, which takes in a parameter, `userChoice`. It also includes a program code segment showing where `filter` is called.
- The response describes what `filter` does and how it contributes to the overall functionality of the program: "filtering one-by-one through my list in order to output a filtered list that matches the user input."

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

- The response includes a program code segment of a student-developed algorithm that includes sequencing, selection (`if` statement), and iteration (`for` loop).
- The response explains in detail how the algorithm works, including that it is "filtering my quotes, sorting them into an empty list, and then outputting that list." It includes details so that someone else could create it: "creating an empty list called `filteredList` (line 14) which will be used to collect the desired quote type. I also created a blank variable called `output` (line 15) which is used in tandem with `filteredList` to create a neat display when the list is output. In order to filter the list, a `for` loop is used (line 16) which traverses list `quoteType`. The `if` statement on the next line (line 17) checks to see if the quote type matches the user selection. If they do match, the item is appended to our empty list, `filteredList` (line 18). Next, on line 19, my variable `output` is defined, which includes adding a line break for better display on the user interface. The output is then displayed onto the screen by using `setText` on line 22."

Row 6:
The response did not earn the point for this row. The response did not meet one of the three criteria:

- Even though the response identifies two distinct calls to the procedure `filter`, each with a different parameter, these parameters do not cause uniquely different paths to the program code to be executed. The code in the `if` statement is executed for both parameter values, and the `if` statement code is never bypassed.
- The conditions that are being tested are if the "user choice calls for movie lines or song lyrics."
- When the parameter is "movie lines"—"movie lines are selected and added to my empty list named `filteredList`". When the parameter is "song lyrics"—"song lyrics are selected and added to my empty list named `filteredList`."

Create Performance Task (continued)

Sample: H

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

Row 1:
The response did not earn the point for this row. The response only met five of the six criteria.

- The video demonstrates the input for the program, showing mouse clicks on buttons in the app.
- The video demonstrates program functionality, showing the user clicking various buttons and getting answers to questions about female legislators.
- The video demonstrates output by showing textual answers to various questions selected by the user.
- The response does not indicate the purpose of the program by describing the problem being solved or the creative interest being pursued. Instead, the purpose given in the response states the program functionality, which is "to display the user some specific information on female legislators".
- The response does describe the functionality of the program, as shown in the video. It gives an example, where "the user chooses the button labeled State - Most Representative on the homepage and then chooses a category such as Female Senate Republicans, the program will display Minnesota."
- The response states that the inputs are "three buttons … which are inputs," and "the user will select something in the dropdown [menu]," and the outputs are "the corresponding screen" and "the answer" based on the dropdown.

Row 2:
The response did not earn the point for this row.

- Although the response gives two code segments, one which shows a list, `states`, being initialized with data from a table, and another that shows the same list being used (`states[i]`), the name of the variable identified as the list is `Female State Legislators`, which isn't the list being shown in the code segments.
- The response describes the contents of the `Female State Legislators` table instead of the list in the code segment (`states`), by stating that "This list contains all of the information in order to provide the user with the answer they are looking for based on what they input into the app."

Create Performance Task (continued)

Row 3:
The response did not earn the point for this row.

- The response includes a code segment that shows a list (`states`) being used to manage complexity in the program. The `states` list holds all the U.S. state information from the State column of the `Female State Legislators` table, which can be of arbitrary length and provide easy access by index.
- The response, however, does not explain why the list represented in the code segment (`states`) manages complexity. Instead, it attempts to explain how the named, selected list, `Female State Legislators`, manages complexity. The response explains that the: "`Female State Legislators` list manages complexity because it allows for the user to choose between many different categories and states while also controlling what the user inputs into the app." But `Female State Legislators` is not managed by this program; it is just used to load the `states` list. The response also explains: "If the program did not contain the list, then the app would be much more complex because it would have to take into account all of the possibilities the user can input, which can even be a thousand." This generic explanation doesn't provide how the program would be rewritten, just that it would need to account for thousands of possibilities.

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

- The response shows a code segment consisting of a student-developed procedure, `housevssenate`, with an explicit parameter, `input`. The parameter has an effect on the procedure since it is used in the first `if` statement inside the loop to set the cumulative totals. The second code segment shows a call to the procedure in the `onEvent` function. The call to the procedure does not need to be in another student-defined procedure.
- The response describes what the procedure does by stating that "It takes into account the state the user selects in order to give the result" and that its contribution to the overall functionality is that the "selected procedure states whether the House or the Senate has more female representatives from 1981 to 2019 in the list."

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

- The response shows a student-developed algorithm that includes sequencing, selection with several `if` statements, and iteration with a `for` loop.
- The response includes detailed steps to allow someone else to recreate the algorithm. The response states the steps as follows: "Two variables were created that stated the value for the Female Senate Total (`cumulativeSenate`) and Female House Total (`cumulativeHouse`) were starting out as zero. The program then includes a `for` loop. This ensures that the name of the state chosen by the user fits the program's appropriate length. Then, the `if` statement, which is in the `for` loop, adds up the Female Senate Total and Female House Total columns separately. After this, the `if else-if else` statement decides whether the sum of the Female Senate Total column is greater than the Female House Total column and vice versa. Based on this, the app tells the user that the Senate or House has more female representation or that there is equal representation."

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 identified procedure that pass different arguments, `Arizona` and `Delaware`, that cause different segments of code to execute.
- In response 3.d.iii, the response describes the first case of `Arizona` where the numbers of representatives are not equal, stating, "it will realize the number in the Female House Total is larger than the Female Senate Total". It further describes the second case of selecting `Delaware` where the number of representatives is equal, stating, "it will come to the conclusion that both of the values in the columns are equal."
- The response describes the different result of each call. In the case of `Arizona`, "the word House will display for the user to show that there is more female representation in the House versus the Senate for that particular state in between 1981 and 2019." In the case of `Delaware`, "it will display Equal Representation for the user."

Create Performance Task (continued)

Sample: I

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

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

- The video demonstrates the program receiving user input in response to 5 questions the squirrel asks (output), and then presents the final score earned by the user at the end (output). The game begins with a question, and after each user input, another message is output as a talking bubble for the squirrel (output). The functionality is the asking of questions by the program and the answering of questions by the user. This satisfies the first three criteria for the video.
- The response does not describe the program's purpose by providing either the problem being solved or the creative interest being pursued through the program. Rather, the response describes the general functionality of the program by stating it is to "create a questionnaire where questions are asked at random to the user and they are promoted to answer it."
- The response further describes the functionality as "questions being asked have two different things that can happen depending on the different answers the user may provide. For example, if the user were to give a correct answer, they would move on and a point will be added to their score, but if they get the answer wrong the point will not be rewarded to the user. At the end of the questionnaire, the user is shown their score"
- The response describes the input and output as, "Input coming from the answers being given by the user and output when being told overall score/progression in the game."

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

- Two distinct code segments are provided: one showing user answer options that were stored to list "`Answers Option 1`" and a second one showing the use of list "`Answers Option 1`" to compare the user input values to the options available in list "`Answers Option 1`" in order to fulfill the program's purpose.
- The name of the list is identified as "`Answers Option 1`."
- The response states that the data being stored is the allowable "answers to modify the points awarded to the user after they input one of the answers."

Create Performance Task (continued)

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

- The response includes a program code segment that shows the `"Answer Options 1"` list being used to manage complexity in the program.
- The response explains how the list `"Answer Options 1"` manages complexity in the program. It states, "I used a list in order to take as many answer possibilities as possible while still maintaining a balance of right and wrong. This is inclusive of spaces between words and capitalization in the program." The response also explains how it could be written differently, "the only way that multiple answers can be accepted in this manner is if you use the list to assign the answer to multiple variables."

Row 4:
The response did not earn the point for this row. The response met only one of the two criteria:

- The program code segment provided in response 3c, "`when Flag clicked`," is not a student-developed procedure and does not contain a parameter. Without a student-developed code segment, the call to the procedure was not met.
- The response accurately describes what the "`when Flag clicked`" code segment does by saying that it identifies "the decision of the user to participate in the trivia game or not."

Row 5:
The response did not earn the point for this row. The response met only one of the two criteria:

- The response includes a program code segment of a student-developed algorithm that includes sequencing, selection (`if answer = yes or answer = Yes then`), but no iteration.
- The response explains how the algorithm works so it can be recreated, "I set 4 different versions of yes and no including the capitalized versions of the words so they will still be accepted if typed. Then with these variables, I was able to dictate through code whether or not the user wanted to play by only allowing them to move on if they typed, 'Yes' or 'yes'. Alternatively, if the user were to answer 'No' or 'no' the code would broadcast to end the procedure with the message 'Ok, have a great day :P'."

Row 6:
The response did not earn the point for this row.

- The given procedure does not use an explicit parameter nor an implicit parameter. Therefore the identified values "yes" and "no" are not considered argument values for the procedure identified in 3c. The conditions being tested are not related to a parameter, and the results are also not related to a parameter.

Create Performance Task (continued)

Sample: J

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

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

- The video shows input of the numerical choices of the menu and the locations being tested to find the villain.
- The video shows the program functioning to allow the user to try to find the villain at various locations.
- The video shows output, congratulating the user when the villain is found and telling the user to try again when the user fails to find the villain.
- The response describes the purpose of the program, which is "to entertain the user as well to teach certain objects that are related to the topic."
- The response describes the functionality of the program, which "works by having user input which would select an option and from those options it would give the user a scenery and that option has a secret list that contains the items/ objects .. etc. which the user would need to guess to find the villain."
- The response describes the input as "what item they are looking for and also what option they want" and the output as "a winning quote or a losing quote."

Row 2:
The response did not earn the point for this row. The response met only one of the three criteria:

- The response includes two program segments, the first showing multiple lists being initialized and the second segment showing another list `list_list` being created using the initial lists. The response does not include a code segment demonstrating the list being used.
- The response does not identify the name of a variable of any of the lists to indicate which list is being discussed.
- The response does describe the data in the lists in general: "The data within the lists will give you objects, buildings, planets, for the scenery that matches with the option you chose." However, it does not identify a specific list and what data is stored in that list.

Create Performance Task (continued)

Row 3:
The response did not earn the point for this row. The response does not meet either of the criteria:

- The program code segment shows multiple lists being used to store data in the program, but there is no program code demonstrating how the list is being used to manage complexity.
- The response does not name a specific list, nor does it explain how the list manages complexity clearly, stating, "The list complexes the program due to all of the elements within the list would need to be used until gaining five points or missing and gaining three points into strikes." The response also does not explain clearly how the program cannot be written without the list or written differently, stating that "the elements would separately give you if else statements that would make your code extremely long and would be a tedious task." The response does not indicate how these `if/else` statements would need to be coded to achieve the same result.

Row 4:
The response did not earn the point for this row. The response met only one of the two criteria:

- The response shows two code student-developed segments. The first code segment is the entire program, which acts as a main function, and the second code segment is a single call to launch the main program. There is no explicit parameter as well.
- The response does describe what the program does by stating that it " is what the program needs to be able to use the options as well to see how many points will give you a winning ending or a losing ending." But the response does not describe how the procedure contributes to the overall functionality since the procedure is the entire program: "This procedure contributes to the function of the program by being the backbone of the entire program and holding everything."

Row 5:
The response did not earn the point for this row. The response met only one of the two criteria:

- The response includes a student-developed algorithm that includes sequencing, selection through the use of nested `if` statements, and iteration through the use of an infinite "`while true`" loop.
- The response does not give enough detail to allow someone else to recreate it. It mentions the main elements of the algorithm: a "point system," "three ways to end the program," "if any element is typed in and it is not in one of the elements within the five list," and "the menu is also what needs to be printed." But it does not give specifics to allow someone else to code this algorithm.

Row 6:
The response did not earn the point for this row.

- The identified procedure does not have an explicit or implicit parameter, so the response does not indicate arguments that would cause different segments of code to execute. Instead, the response indicates that the first call "starts the program" and the second call "will give the options … and start the game part of the program." The conditions being tested and the results of each call do not correspond to the use of a parameter for a procedure. These responses only indicate how the program runs.