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 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 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.