3. WRITTEN RESPONSES (CREATED INDEPENDENTLY)

Submit your responses to prompts 3a – 3d, which are described below. Your response to all prompts combined must not exceed 750 words (program code is not included in the word count). Collaboration is **not** allowed on the written responses. Instructions for submitting your written responses are available on the **AP Computer Science Principles Exam Page** on AP Central.

3 a. Provide a written response that does all three of the following:

Approx. 150 words (for all subparts of 3a combined)

i. Describes the overall purpose of the program

ii. Describes what functionality of the program is demonstrated in the video

iii. Describes the input and output of the program demonstrated in the video

3 b. Capture and paste two program code segments you developed during the administration of this task that contain a list (or other collection type) being used to manage complexity in your program.

Approx. 200 words (for all subparts of 3b combined, exclusive of program code)

i. The first program code segment must show how data have been stored in the list.

ii. The second program code segment must show the data in the same list being used, such as creating new data from the existing data or accessing multiple elements in the list, as part of fulfilling the program's purpose.

Then, provide a written response that does all three of the following:

iii. Identifies the name of the list being used in this response

iv. Describes what the data contained in the list represent in your program

v. Explains how the selected list manages complexity in your program code by explaining why your program code could not be written, or how it would be written differently, if you did not use the list

DEFINITION:

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 may be referred to by different names, such as **arrays**, depending on the programming language.

DEFINITION:

Collection Type

A **collection type** is a type that aggregates elements in a single structure. Some examples include lists, databases, hash tables, dictionaries, and sets.

IMPORTANT:

The data abstraction must make the program easier to develop (alternatives would be more complex) or easier to maintain (future changes to the size of the list would otherwise require significant modifications to the program code).

3 c. Capture and paste two program code segments you developed during the administration of this task that contain a student-developed procedure that implements an algorithm used in your program and a call to that procedure.

Approx. 200 words (for all subparts of 3c combined, exclusive of program code)

IMPORTANT:

Built-in or existing procedures and language structures, such as event handlers and main methods, are not considered student-developed.

i. The first program code segment must be a student-developed procedure that:

- ☐ Defines the procedure's name and return type (if necessary)
- ☐ Contains and uses one or more parameters that have an effect on the functionality of the procedure
- ☐ Implements an algorithm that includes sequencing, selection, and iteration

ii. The second program code segment must show where your student-developed procedure is being called in your program.

Then, provide a written response that does both of the following:

iii. Describes in general what the identified procedure does and how it contributes to the overall functionality of the program

iv. Explains in detailed steps how the algorithm implemented in the identified procedure works. Your explanation must be detailed enough for someone else to recreate it.

3 d. Provide a written response that does all three of the following:

Approx. 200 words (for all subparts of 3d combined)

i. Describes two calls to the 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.

First call:

Second call:

ii. Describes what condition(s) is being tested by each call to the procedure

Condition(s) tested by the first call:

Condition(s) tested by the second call:

iii. Identifies the result of each call

Result of the first call:

Result of the second call: