**3a.** Provide a written response that:

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

1. Describes the overall purpose of the program;

|  |
| --- |
| Purpose of the program is to help the user decide whether or not to do something |

1. Describes what functionality the video illustrates;

|  |
| --- |
| When the user clicks on the screen, the magic 8 ball appears to make a recommendation and the icons on the screen change to represent if it's a positive, neutral, or negative response. |

1. Describes the input and output of the program shown in the video

|  |
| --- |
| Input is the user clicking on the screen. Output is the text displayed on the magic 8 ball and the different icons. |

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

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

1. The first program code segment must show how data has been stored in the list.

|  |
| --- |
|  |

1. The second program code segment must show the data in the same list being processed, 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:

1. Identifies the name of the list being processed in this response

|  |
| --- |
| Name of list = answers |

1. Describes what the data contained in the list is representing in your program

|  |
| --- |
| List of strings which store responses randomly chosen to display on the screen. |

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

|  |
| --- |
| Manages complexity because my code would be longer without a list. |

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

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

1. This 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; and
* Implements an algorithm that includes sequencing, selection and iteration.

|  |
| --- |
|  |

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

|  |
| --- |
|  |

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

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

|  |
| --- |
| The function setImages controls what icons are displayed on the screen after a random response is chosen. This helps the user know if the response was positive or not. |

1. Explains in detailed steps how the algorithm implemented in the selected procedure accomplishes its task. Your explanation must be detailed enough for someone else to recreate it.

|  |
| --- |
| The function setImages works by choosing an image and calling another function to set the color of the image. |

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

*(Approx. 200 words, for all subparts of 3d combined, excluding program code)*

1. Describes two calls to the selected procedure identified in written response 3c. Each call must pass different arguments that cause a different segment of code in the algorithm to execute;

First call:

|  |
| --- |
| Call # 1: setImages(2) |

Second call:

|  |
| --- |
| Call #2: setImages(7) |

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

Condition(s) tested by the first call:

|  |
| --- |
| The list is organized so the first three elements are positive responses, the next three are neutral, and the last three are negative. This call is passing through the argument 2. Lines 26-27 check if 2 is less than three. |

Condition(s) tested by the second call:

|  |
| --- |
| This call is passing through the argument 7. Lines 26-29 are skipped, because 7 not less than 3 and 7 is not less than 6. Lines 30-32 run because it is the final branch of the if else statement. |

1. Identifies the result of each call.

Result of the first call:

|  |
| --- |
| 2 is less than 3 is true, so the response is positive and the styleImages function is run to set the icon to a yellow star. |

Result of the second call:

|  |
| --- |
| The response is negative and therefore the styleImages function is run to set the icon to a red crossed out icon. |