

---

# AP<sup>®</sup> Computer Science Principles

## Sample Student Responses and Scoring Commentary Set 2

### **Inside:**

#### **Written Response 2**

- ☒ **Scoring Guidelines**
- ☒ **Student Samples**
- ☒ **Scoring Commentary**

**Digital Portfolio Components provided separately**

**Written Response 2****3 points****General Scoring Notes**

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

Reporting Category	Scoring Criteria	Decision Rules
<b>Written Response 2(a): Algorithm Development</b>  <b>(0–1 points)</b>	<p>The written response:</p> <ul style="list-style-type: none"> <li>• describes the conditional statement, including its Boolean expression.</li> <li>• describes what the program <i>code segment</i> inside the conditional statement does in general when the Boolean expression of the conditional statement is <i>false</i>.</li> </ul>	<p><b>Consider the Personalized Project Reference and Written Response 2(a) when scoring this point.</b></p> <ul style="list-style-type: none"> <li>• If multiple conditional statements are included in the Procedure section of the Personalized Project Reference, use the first conditional statement to determine whether the point is earned.</li> <li>• The conditional statement can be found in either part (i) or part (ii) of the Procedure section of the Personalized Project Reference.</li> <li>• The conditional statement does not need to be contained in a procedure to earn this point.</li> <li>• The response does not have to explicitly state the Boolean expression as long as it is described.</li> <li>• The response may earn this point for a conditional statement that either does or does not contain an else clause.</li> </ul> <p><b>Do NOT award a point if any one or more of the following is true:</b></p> <ul style="list-style-type: none"> <li>• The Procedure section of the Personalized Project Reference does not contain a conditional statement.</li> <li>• The description of the Boolean expression does not match the code in the first conditional statement.</li> <li>• The description of the behavior of the code when the expression is <i>false</i> does not match the code in the first conditional statement.</li> <li>• The response only recites lines of code instead of providing a general description.</li> <li>• The response describes a conditional statement or behavior that is implausible, inaccurate, or inconsistent with the program.</li> </ul>

Reporting Category	Scoring Criteria	Decision Rules
<p><b>Written Response 2(b): Errors and Testing</b></p> <p><b>(0–1 points)</b></p>	<p>The written response:</p> <ul style="list-style-type: none"> <li>describes the outcome of the procedure call based on the <i>argument(s)</i>, if any, used in the call.</li> </ul> <p>AND</p> <ul style="list-style-type: none"> <li>includes a procedure call with at least one different argument value that will produce the same outcome.</li> <li>explains why this call produces the same outcome.</li> </ul> <p>OR</p> <ul style="list-style-type: none"> <li>explains why it is not possible to write a new procedure call with at least one different argument value that produces the same outcome.</li> </ul>	<p><b>Consider the Personalized Project Reference and Written Response 2(b) when scoring this point.</b></p> <ul style="list-style-type: none"> <li>If multiple procedures are included in part (i) of the Procedure section of the Personalized Project Reference:             <ul style="list-style-type: none"> <li>Use the procedure identified in the written response to determine whether the point is earned.</li> <li>If no procedure is identified in the written response, then use the first procedure to determine whether the point is earned.</li> </ul> </li> <li>A procedure with explicit parameters, implicit parameters, or no parameters can earn this point if scoring guidelines are met.</li> <li>If the procedure has explicit or implicit parameters, the procedure call must include appropriate argument(s) that will lead to the described outcome or behavior.</li> <li>The syntax of the procedure call does not need to be correct as long as it identifies the correct arguments to the procedure, if necessary.</li> </ul> <p><b>Do NOT award a point if any one or more of the following is true:</b></p> <ul style="list-style-type: none"> <li>A procedure is not identified in part (i) of the Procedure section of the Personalized Project Reference.</li> <li>A procedure call is not included in part (ii) of the Procedure section of the Personalized Project Reference.</li> <li>The response does not apply to the procedure in part (i) or the procedure call in part (ii) of the Procedure section of the Personalized Project Reference.</li> <li>The response includes an explanation that is implausible, inaccurate, or inconsistent with the procedure and its given argument(s), if any.</li> </ul>

Reporting Category	Scoring Criteria	Decision Rules
<b>Written Response 2(c): Data and Procedural Abstraction</b>  <b>(0–1 points)</b>	<p>The written response:</p> <ul style="list-style-type: none"> <li>identifies the <i>parameter(s)</i> of the procedure.</li> <li>explains how the identified parameter(s) use abstraction to manage complexity in their program.</li> </ul>	<p><b>Consider the Personalized Project Reference and Written Response 2(c) when scoring this point.</b></p> <ul style="list-style-type: none"> <li>If multiple procedures are included in part (i) of the Procedure section of the Personalized Project Reference, use the first procedure to determine whether the point is earned.</li> <li>The parameter(s) used in the procedure must be explicit. Explicit parameters are defined in the header of the procedure.</li> </ul> <p><b>Do NOT award a point if any one or more of the following is true:</b></p> <ul style="list-style-type: none"> <li>A procedure is not identified in part (i) of the Procedure section of the Personalized Project Reference.</li> <li>The response does not apply to the procedure in part (i) of the Procedure section of the Personalized Project Reference.</li> <li>The response identifies arguments instead of parameters for the first scoring criterion.</li> <li>The procedure identified in part (i) of the Procedure section of the Personalized Project Reference does not include at least one explicit parameter.</li> <li>The use of any of the parameters is irrelevant (i.e., does not affect the outcome of the procedure or is reassigned immediately before being used).</li> <li>The response includes an explanation that is implausible, inaccurate, or inconsistent with the procedure.</li> <li>The procedure is not a student-developed procedure (e.g. an event handler).</li> </ul>

## AP Computer Science Principles Create Performance Task Terminology

**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.

**Arguments:** The values of the parameters when a procedure is called.

**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.

**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.

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

**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.

**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.

**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.

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

**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.

**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.

**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. A procedure is executed through the use of a procedure call.

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

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

**Selection / conditional statement:** A selection / conditional statement affects the sequential flow of control by executing different statements based on a condition being true or false. The use of if-statements and try / exception statements are examples of selection / conditional statements.

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

**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.”

2(a)

The first conditional statement of my procedure determines whether a student will receive an 'A' in a class. If the value returned by the previous calculation, which is the average of the student's assignment grades, is greater than 89, the statement evaluates to true, and the variable 'lettergrade' is assigned a value of 'A'. If this conditional statement evaluates to false, the variable 'lettergrade' will not be an 'A', and the procedure continues running. It will continue checking the average grade to determine whether the student will receive a B, C, D, or F.

2(b)

The procedure is designed to find two new variables, 'avg' and 'lettergrade.' It will accept a list of values, from which the average of the values will be determined. Then, based on the standard grading scale, a letter will be assigned to the average. For example, you could call the procedure with the parameter, '(findavg([90, 100, 80]))'. The procedure would determine that the average of the list is 90, assigning an 'A' to the variable 'lettergrade'. To receive the same outcome for both variables, another list of numbers that also average to 90 would be needed. For example, the list '(findavg([70, 90, 110]))' would return the same results, since the average of the three numbers is still 90.

2(c)

My procedure uses a list as its parameter. It receives a list of values from the user and outputs the average of these values. This parameter uses abstraction because it can be used for many different values at once, and can be called multiple times. The procedure will still function when different lists are input, and it can be called multiple times. This means that this long chunk of code only needs to be written once, saving time and managing complexity. The parameter being a list is also helpful, because without it many separate variables would need to be input into the procedure, which would require more code and complexity. The list manages this by storing all of the values in one place.

2(a)

My conditional statements in the procedure compare the chosen due date of an assignment to the current day in order to display warnings such as "due today" or "late". The first conditional compares the year of the due date to the current year, where the due date year is a parameter and the current year is a global variable. If the years are the same and the conditional statement evaluates to true, then the procedure will continue to another conditional that compares the due date day and month to the current day and month. If the due date day and month are the exact same as the current day and month, this means the due date is today and the warning variable will be set to "due today". If the months are the same but due day is smaller than the current day, or if the due month is smaller than the current month, it means the due date has passed and the warning will be set to "late". If the months are the same but the due day is greater than the current day by only one, it means that the due date is tomorrow, so the warning will be set to "due tomorrow". If none of these conditionals are true, then there is no warning set. Going back to the first conditional that compares the years, if it evaluates to false then it continues the conditional to compare if the due date year is past the current year, and if it is then the warning will be set to "late". Lastly, the first conditional checks if the due date year is greater than the current year, and if it is then no warning is set. Overall, the entire conditional will set the warning to a global variable as a string, and once the procedure is called, the textbox next to the due date will display this warning variable.

2(b)

The expected outcome that the procedure should produce is a warning next to the chosen due date if it needs it. The procedure takes three arguments: day, month, and time, which come from the due date. These arguments are evaluated to produce a certain warning next to an assignment. If there were a new procedure call with the argument value of day changed, it would most likely produce the same outcome. For example, if in one call the current day variable was 15 and the due date day argument was 11 (both dates are on the same month and year), the procedure would set the warning to "late" because the due date passed. If this new procedure call had the argument value for day changed to 13 instead of 11 (current day is the same), the procedure would still produce the same outcome, which is a warning set to "late". Both calls produce the same warning because both due dates have passed the current date, even though they are different numbers. The procedure call would look like `compare_dates(DatePicker1Day, DatePicker1Month, DatePicker1Year)` with the value of `DatePicker1Day` being 13.

2(c)

The parameters used in this procedure are the day, month, and year of the due date. These parameters make the code less complex because there are different due dates set in the whole program that need to go through the same procedure to set a warning. If there were no parameters, the whole procedure would have to be copied into each date picker so that the day, month, and year variables of a chosen due date are compared each time the due date is set. With parameters, these three variables can be called in the procedure even though they change for each different due date.

2(a)

The conditional statement says that if the corresponding item to the index (i) in the list taken from the argument is the string "correct", add 1 to the variable totalCorrect, which keeps track of how many items are "correct". If the Boolean expression of the conditional statement evaluates to false, it means that the corresponding item to the index (i) is not "correct", so it won't add 1 to totalCorrect and that piece of code (line 6-8) is repeated until all items in the list have been evaluated to be "correct" or not. Then the value of totalCorrect is returned.

2(b)

The procedure call score(userAnswers) in line 58 is intended to produce the number of questions the user got correct. The procedure call score(userAnswers) in line 59 is intended to produce the number of questions the user got wrong which is why 5, which is the number of questions that the user needs to answer, is subtracted from the value the procedure call produces.

It's not possible to write a new procedure call that produces the same outcome because the argument is entirely based off the user's input of how many questions they got correct which is in the list userAnswers. userAnswers is the only list that keeps track of the user's answers and we can't predict how many questions a user will get correct so we can't set a different argument value for the procedure call.

2(c)

The parameter used for the procedure is "list". The procedure has a parameter so that when I need to use another list to check the amount of answers that are "correct", I don't need to make a new procedure just for that list and I can just insert the other list into that procedure call's argument. This abstraction manages complexity in my program.



2(a)

One conditional statement in the procedure "Find High Score", which has two parameters of list "P1 Scores" and list "P2 Scores", finds the highest value from each list, and then compares the highest value between the two lists to see if the scores mean that "Player 1 won", "Player 2 won", or "It's a tie", is an if statement that, for each item in the list of "P1 Scores", sets the integer variable "p1 high score" to that item if that item is greater than the "p1 high score". If the item at the current index of the list "P1 Scores" is less than the current "p1 high score", then the boolean expression would evaluate to false, and the variable "p1 high score" would not be set to the value of that item.

2(b)

The procedure call is intended to produce the outcome of displaying whether the procedure "Find High Score", which accepts two inputs of list "player 1 scores" and list "player 2 scores", finds the highest value from each list, compares the highest value between the two lists to see if the scores mean that "Player 1 won", "Player 2 won", or "It's a tie". Given that "player 1 scores" is a list of numerical values, and as a part of this procedure, the procedure simply finds the highest value from each list of "player 1 scores" and "player 2 scores" regardless of whether or not there is one more value in each list, one could append any numerical value that does not exceed the data type maximum value for each value in "player 1 scores" and is less than the highest value in the list to the next, and last, index of "player 1 scores", and set the new list to the list "player 1 scores edited", before inputting "player 1 scores edited" in the place of "player 1 scores" in the given procedure call, and the final output will be the same.

2(c)

The procedure "Find High Score", which is identifiable from part (i) of the Procedure section of the Personalized Project Reference, has two parameters of list "P1 Scores" and list "P2 Scores". The procedure finds the highest value from each inputted list, and then compares the highest value between the two lists to see if the scores mean that "Player 1 won", "Player 2 won", or "It's a tie".

The parameters "P1 Scores" and "P2 Scores" use abstraction to manage complexity by each storing individual user input of numerical values in a list, making filtering each list in order to find the highest value in each list able to be completed with one, simple "for each" loop for each list. In total, there are two "for each" loops in the procedure.

There is one "for each" loop to evaluate the highest value in the input for the parameter "P1 Scores", which includes an if statement that compares the variable "p1 high score", which is initially set to the first value in the input of the list that the parameter "P1 Scores" accepts, to each value in the input for the parameter "P1 Scores", and then consequently set "p1 high score" to that value if it is greater in order to find the highest value in the input of the list that the parameter "P1 Scores" accepts.

There is another "for each" loop to evaluate the highest value in the input for the parameter "P2 Scores", which includes an if statement that compares the variable "p2 high score", which is initially set to the first value in the input of the list that the parameter "P2 Scores" accepts, to each value in the input for the parameter "P2 Scores", and then consequently set "p2 high score" to that value if it is greater in order to find the highest value in the input of the list that the parameter "P2 Scores" accepts.

In this way, the procedure does not need to accept each value that was entered by a user as an individual input, making the program as a whole possibly easier to understand and minimizing unnecessary code.

2(a)

My conditional statement analyzes the list and data set "year" that is in the data table. If the year that the user selects from the dropdown is equal to the year in the list, the code will display the information about the winner from the winner list and the number of teams participating from the teams list in the correlating text box. If the "years" section from the dropdown does not match the "year" in the list or if the user does not select a year from the dropdown, the conditional statement will turn to false. This then means no data will be written in the outbox and the conditional statement will not run.

2(b)

My procedure is supposed to analyse the year that the user selects from the dropdown, and then find that given year in order to set the outboxes using specific data from the same row that the year is (teams and winner). If you wanted to write a new procedure call, you could change the name of the function or name of the variable. For example, instead of function `worldcup(years)` in line 35, you could change it to function `outcome(years)` and then change it in line 32 to `outcome(user)` instead of `worldcup(user)`. You could also change `var cupyear= year.length` in line 36 to `var specificyear = year.length` and then replace "cupyear" in line 37 to "specificyear". These changes will produce the same outcome because it is changing the names and keeping them consistent with one another without altering the names of the data sets in the tables. These data sets include: "year" "teams" and "winner".

2(c)

The parameters that were used in my procedure are the for/if parts, shown in lines 37 and 38. These add complexity to my program because they are analysing the lists to get the correct data. By having these parameters here, the code segment can read the list and sort through the information quickly. These add abstraction because the code will only run if the year is shown in the list. It then continues to read the data to make sure everything is correct when it is inserted in the outbox. unnecessary code.

2(a)

My first conditonal statement is found in the "def difficulty\_level(level)" function. It is divided into 3 "if-statements" which are dependent on the user's input. If level == "easy" it evaultes to "true" (which randomly generates a country from the easy level), but if it is not equal, it evalautes to "false". The "if level == "easy" is the first Boolean expression of the conditional statement of the program. This happens for the other two if-statements as well. If the statement evaluated to "false" for the first expression it will be reevaluated for level == "medium", if it is true, and the two statments are equal, it will randomly generate a country for the medium level. If it was false it will be reevaluated for the "hard" level. However if it does not equal "hard" as well the program will error. If I were to create this program again I would make sure to include an else-statement which would allow the user to input the difficulty level again until their input is valid.

2(b)

The procedure call intends for the user to choose which level they prefer. This is done with the previous line which asks the user to input either "easy", "medium", or "hard". After the "level" parameter is set to one of these three inputs, it displays a welcome statement to that level and prints the first question/fact (by generating a random country from that level). It is not possible to write a new procedure call with at least one different argument without changing the other aspects of the code. If we were to change the "level" argument it would not be possible to call the function "def difficulty\_level(level):" since the function requires the parameter to be set to some value. The only way to obtain the same outcome without altering the code excessively would be to replace the call: "difficulty\_level(level)" with the direct input to be "difficulty\_level(input("Choose your difficulty: 'easy', 'medium', or 'hard' "))". However, this would then need to remove the parameter entirely for the "def difficulty\_level" function so that it would become "def difficulty\_level()".

2(c)

The parameter used in the procedure "def difficulty\_level" is "level". This parameter allows abstraction since it simplifies the program. Without the parameter the code would no longer work since the difficulty level is dependent on what the user inputs for "level" which is set as the parameter. The parameter allows us to create a singular function that uses if-statements which will decide the difficulty level. If we were to remove the parameter and replace it with a statement asking for the user's input (input("Choose your difficulty: 'easy', 'medium', or 'hard' ")) we would not have any variable to compare the input options to.

The parameter is also useful since it allows us to call the function in the play\_again function. If the parameter was not included the code would have to be rewritten so that the whole function looped over again. This would add much more complexity to the program since I would have to rewrite the entire code underneath a loop. The parameter allows me to break down the code into multiple different functions which helps manage complexity and reduces the chance of errors.

## Question 2

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

### Overview

Responses to this question were expected to demonstrate that the student could:

- explain how a code segment functions, including evaluating expressions that use logic and relational operators and determining the result of a conditional statement. (Written response 2(a): Algorithm Development),
- identify inputs and corresponding expected outputs or behaviors that can be used to check the correctness of an algorithm or program (Written response 2(b): Errors and Testing), and
- explain how the use of procedural abstraction manages complexity in a program (Written response 2(c): Data and Procedural Abstraction).

Written response 2(a) asked students to describe their conditional statement including its Boolean expression and what the procedure does in general when the Boolean expression is false. Students were required to identify a conditional statement (i.e. a selection statement), evaluate its Boolean expression, and explain how it functions in one of two cases.

Written response 2(b) asked students to describe the outcome of the procedure call they identified in the Personalized Project Reference, demonstrating that they could identify the inputs and corresponding outputs of a procedure call. It then asked students to write a new procedure call with at least one different argument that produces the same outcome (or to state why it was not possible to do so). This ability is critical to identifying and correcting errors in code in two ways. First, for a student to determine if their procedure is functioning correctly, the student must first understand the procedure's expected behavior for a given input so that they can match the expected output to the observed output when they run the procedure. Second, it is important to recognize when two different tests produce the same behavior, or to recognize that no two calls can produce the same behavior, to ensure that all test cases do not test the same functionality of the procedure.

Part (c) asked students to identify the parameters of their procedure and explain how these parameters use abstraction to manage complexity in their program. The use of parameters to generalize the functionality of a procedure is a key aspect of procedural abstraction, allowing a single procedure to be called using a variety of data values to solve multiple instances of the same problem. In this part, students were expected to show that they understood this specific aspect of procedural abstraction, beyond procedural abstraction as a general concept to simply reduce the amount of code written.

## Question 2 (continued)

**Sample: A**

**Score:**

**Question 2(a): 1**

**Question 2(b): 1**

**Question 2(c): 1**

**Question 2(a):**

The response earned this point, meeting both criteria:

- The response describes the conditional statement: “The first conditional statement of my procedure determines whether a student will receive an ‘A’ in a class.” The response also describes its Boolean expression: “If the value returned by the previous calculation, which is the average of the student’s assignment grades, is greater than 89, the statement evaluates to true”.
- The response describes what the program code segment inside the conditional statement does in general when the Boolean expression is false: “If this conditional statement evaluates to false, the variable ‘lettergrade’ will not be an ‘A’, and the procedure continues running. It will continue checking the average grade to determine whether the student will receive a B, C, D, or F.”

**Question 2(b):**

The response earned this point, meeting both criteria:

- The response describes the outcome of the procedure call: “The procedure is designed to find two new variables, ‘avg’ and ‘lettergrade.’ It will accept a list of values, from which the average of the values will be determined.” The response also describes the procedure call and a specific set of argument values: “you could call the procedure with the parameter, ‘(findavg([90, 100, 80]))’ . The procedure would determine that the average of the list is 90, assigning an ‘A’ to the variable ‘lettergrade’ .”
- The response correctly explains why another call produces the same outcome: “To receive the same outcome for both variables, another list of numbers that also average to 90 would be needed. For example, the list ‘(findavg([70, 90, 110]))’ would return the same results, since the average of the three numbers is still 90.”

**Question 2(c):**

The response earned this point, meeting both criteria:

- The response identifies the parameters of the procedure: “My procedure uses a list as its parameter. It receives a list of values from the user.”
- The response explains how the identified parameter(s) use abstraction to manage complexity in the program: “The procedure will still function when different lists are input, and it can be called multiple times. This means that this long chunk of code only needs to be written once, saving time and managing complexity.”

## Question 2 (continued)

**Sample: B**

**Score:**

**Question 2(a): 1**

**Question 2(b): 1**

**Question 2(c): 1**

**Question 2(a):**

The response earned this point, meeting both criteria:

- The response describes the conditional statement: “My conditional statements in the procedure compare the chosen due date of an assignment to the current day in order to display warnings such as ‘due today’ or ‘late’... then the procedure will continue to another conditional that compares the due date day and month to the current day and month.” The response also describes the Boolean expression: “If the years are the same and the conditional statement evaluates to true.”
- The response describes what the program code segment inside the conditional statement does in general when the Boolean expression is false: “Going back to the first conditional that compares the years, if it evaluates to false then it continues the conditional to compare if the due date year is past the current year, and if it is then the warning will be set to ‘late’.”

**Question 2(b):**

The response earned this point, meeting all three criteria:

- The response describes the outcome of the procedure call based on the arguments: “The expected outcome that the procedure should produce is a warning next to the chosen due date if it needs it. The procedure takes three arguments: day, month, and time, which come from the due date.”
- The response describes a new procedure call: “For example, if in one call the current day variable was 15 and the due date day argument was 11.”
- The response explains why the new call produces the same outcome: “(both dates are on the same month and year), the procedure would set the warning to ‘late’ because the due date passed.”

**Question 2(c):**

The response earned this point, meeting both criteria:

- The response identifies the parameters of the procedure: “The parameters used in this procedure are the day, month, and year of the due date.”
- The response explains how the identified parameters use abstraction to manage complexity in the program: “If there were no parameters, the whole procedure would have to be copied into each date picker so that the day, month, and year variables of a chosen due date are compared each time the due date is set.” The response states, “With parameters, these three variables can be called in the procedure even though they change for each different due date.”

## Question 2 (continued)

**Sample: C**

**Score:**

**Question 2(a): 1**

**Question 2(b): 0**

**Question 2(c): 1**

**Question 2(a):**

The response earned this point, meeting both criteria:

- The response describes the conditional statement: “if the corresponding item to the index (i) in the list taken from the argument is the string ‘correct’, add 1 to the variable totalCorrect, which keeps track of how many items are ‘correct.’” The response describes the Boolean expression: “if the corresponding item to the index (i) in the list taken from the argument is the string “correct.””
- The response describes what the program code segment inside the conditional statement does in general when the Boolean expression is false: “it won't add 1 to totalCorrect and that piece of code (line 6-8) is repeated until all items in the list have been evaluated to be ‘correct’ or not.”

**Question 2(b):**

The response did not earn this point, meeting one of the two criteria:

- The response describes the outcome of the procedure call: “The procedure call score(userAnswers) in line 58 is intended to produce the number of questions the user got correct.”
- The response explains why it is not possible to write a new procedure call with at least one different argument value that produces the same outcome: “because the argument is entirely based off the user's input of how many questions they got correct which is in the list userAnswers.” However, it is possible to enter two different lists with the same number of incorrect answers and get the same total correct as the outcome.

**Question 2(c):**

The response earned this point, meeting both criteria:

- The response identifies the parameters of the procedure: “The parameter used for the procedure is ‘list.’”
- The response explains how the identified parameter uses abstraction to manage complexity in the program: “when I need to use another list to check the amount of answers that are ‘correct’, I don't need to make a new procedure just for that list and I can just insert the other list into that procedure call's argument.”



**Question 2 (continued)****Sample: D****Score:****Question 2(a): 1****Question 2(b): 1****Question 2(c): 0****Question 2(a):**

The response earned this point, meeting both criteria:

- The response describes the conditional statement: “One conditional statement in the procedure ‘Find High Score’...finds the highest value from each list.” The response describes the Boolean expression: “if that item is greater than the ‘p1 high score.’”
- The response describes what the program code segment inside the conditional statement does in general when the Boolean expression is false: “If the item at the current index of the list ‘P1 Scores’ is less than the current ‘p1 high score’, then the boolean expression would evaluate to false, and the variable ‘p1 high score’ would not be set to the value of that item.”

**Question 2(b):**

The response earned this point, meeting all three criteria:

- The response describes the outcome of the procedure call: “The procedure call is intended to produce the outcome of displaying whether the procedure ‘Find High Score’, which accepts two inputs of list ‘player 1 scores’ and list ‘player 2 scores’, finds the highest value from each list, compares the highest value between the two lists to see if the scores mean that ‘Player 1 won’, ‘Player 2 won’, or ‘It's a tie.’”
- The response describes the procedure call and argument value: “one could append any numerical value that does not exceed the data type maximum value for each value in ‘player 1 scores’ and is less than the highest value in the list to the next, and last, index of ‘player 1 scores.’” Although this response does not provide a specific value, it does provide a general description of the conditions that will produce the same outcome.
- The response correctly explains why this call produces the same outcome by describing that if the added value is less than the maximum value in the list “the final output will be the same.”

**Question 2(c):**

The response did not earn this point, meeting one of the two criteria:

- The response identifies the parameters of the procedure: “list ‘P1 Scores’ and list ‘P2 Scores.’”
- The response fails to correctly identify how the parameters use abstraction to manage program complexity. The response explains, “The parameters ‘P1 Scores’ and ‘P2 Scores’ use abstraction to manage complexity by each storing individual user input of numerical values in a list, making filtering each list in order to find the highest value in each list able to be completed with one, simple ‘for each’ loop for each list.” In this case, the response addresses the complexity of the list and conditional statements rather than correctly addressing how the parameters generalize the procedure by allowing for various lists to be processed by different procedure calls.

## Question 2 (continued)

**Sample: E**

**Score:**

**Question 2(a): 1**

**Question 2(b): 0**

**Question 2(c): 0**

**Question 2(a):**

The response earned this point, meeting both criteria:

- The response describes the conditional statement: “My conditional statement analyzes the list and data set ‘year.’” The response describes the Boolean expression: “If the year that the user selects from the dropdown is equal to the year in the list, the code will display the information about the winner from the winner list.”
- The response describes what the program code segment inside the conditional statement does in general when the Boolean expression is false: “If the ‘years’ section from the dropdown does not match the ‘year’ in the list... the conditional statement will turn to false. This then means no data will be written in the outbox.”

**Question 2(b):**

The response did not earn this point, meeting one of the three criteria:

- The response describes the outcome of the procedure call based on the year entered by the user. The response states that the procedure will “set the outboxes using specific data from the same row that the year is (teams and winner).”
- The response does not describe a new procedure call with a different argument. Instead, it describes the rewriting of the procedure in part (i) to get the same outcome. The response states, “you could change it to function outcome(years) and then change it in line 32 to outcome(user) instead of worldcup(user).” It does not describe a different argument that would give the same outcome.
- The response explains why this call produces the same outcome: “These changes will produce the same outcome because it is changing the names.” However, this change is not due to the use of a different argument value.

**Question 2(c):**

The response did not earn this point, meeting neither of the two criteria:

- The response does not identify the parameters of the procedure. The response states, “The parameters that were used in my procedure are the for/if parts, shown in lines 37 and 38,” but it does not identify the parameters in the code.
- The response inaccurately explains how the parameters use abstraction to manage complexity in the program. The response explains, “These add abstraction because the code will only run if the year is shown in the list.” This does not describe the management of complexity.

**Question 2 (continued)****Sample: F****Score:****Question 2(a): 1****Question 2(b): 0****Question 2(c): 0****Question 2(a):**

The response earned this point, meeting both criteria:

- The response describes the conditional statement, including its Boolean expression. The response describes the conditional statement as: “My first conditonal statement is found in the “def difficulty\_level(level)” function. It is divided into 3 “if-statements” which are dependent on the user’s input.” The response describes the Boolean expression as: “If level == ‘easy’ it evaultes to “true” (which randomly generates a country from the easy level), but if it is not equal, it evalautes to “false”.”
- The response describes what the code segment inside the conditional statement does in general when the Boolean expression is false by stating, “If the statement evaluated to “false” for the first expression it will be reevaluated for level == “medium”, if it is true, and the two statments are equal, it will randomly generate a country for the medium level.”

**Question 2(b):**

The response did not earn this point, meeting one of two criteria:

- The response describes the outcome of the procedure call: “it displays a welcome statement to that level and prints the first question/fact (by generating a random country from that level).”
- The response explains that it is not possible to write a new procedure call with at least one different argument value that produces the same outcome. However, since the response uses a variable as the argument in the procedure call and the response does not provide a specific value and associated output, there is no reference from which to explain why no other value will produce the same output.

**Question 2(c):**

The response did not earn this point, meeting one of the two of the criteria:

- The response identifies the parameter of the procedure: “level.”
- The response does not explain how the parameter manages complexity in the program code. Rather, it explains what adjustments would need to be made if the parameter was removed.