

---

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

## Sample Student Responses and Scoring Commentary Set 1

### **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 2a: Algorithm Development</b>  <b>(0–1 points)</b>	<p>The written response:</p> <ul style="list-style-type: none"> <li>• identifies the Boolean expression in the first selection statement.</li> <li>• identifies a specific value or set of values that will cause the Boolean expression of the selection statement to evaluate to <code>true</code>.</li> <li>• explains why the specified value or set of values will cause the expression to evaluate to <code>true</code>.</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 selection statements are included in the Procedure section of the Personalized Project Reference, use the first selection statement to determine whether the point is earned.</li> <li>• The selection 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 identified Boolean expression can include the header of the selection statement (e.g., <code>if (x &gt; 2)</code> instead of <code>x &gt; 2</code>).</li> <li>• The response may earn this point for a selection statement that either does or does not contain an <code>else</code> 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 selection statement.</li> <li>• The identified Boolean expression does not match the code in the first selection statement.</li> <li>• The identified value or set of values that will cause the expression to evaluate to <code>true</code> does not match the code in the first selection statement.</li> <li>• The response identifies a Boolean expression or value (or set of values) that is implausible, inaccurate, or inconsistent with the program.</li> </ul>

Reporting Category	Scoring Criteria	Decision Rules
<b>Written Response 2b: Errors and Testing</b>  <b>(0–1 points)</b>	<p>The written response:</p> <ul style="list-style-type: none"> <li>describes a modification another programmer could make that would cause the <i>procedure</i> to have a logic error.</li> <li>describes how the behavior of the procedure would change because of this introduced error.</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>The modification can be changing code, adding code, and/or deleting code.</li> <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 referenced in the written response to determine whether the point is earned.</li> <li>If no procedure is referenced in the written response, then use the first procedure to determine whether the point is earned.</li> </ul> </li> <li>A logic error is a mistake in an algorithm or program that causes it to behave incorrectly or unexpectedly. Responses that describe a modification that would lead to the program crashing may earn this point.</li> <li>The response may describe more than one modification that leads to a logic error. If the response describes more than one modification, the change in behavior must be described correctly for every modification.</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 included 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 states only that the program will not run without further explanation.</li> <li>The response describes a modification or change in behavior that is implausible, inaccurate, or inconsistent with the procedure or the described modification.</li> </ul>

<b>Written Response 2c: Data and Procedural Abstraction</b>  <b>(0–1 points)</b>	<p>The written response:</p> <ul style="list-style-type: none"><li>explains how the <i>code segment</i> that accesses or manipulates <i>data stored in the list</i> (or other <i>collection type</i>) would need to change if another programmer added several new elements to the end of the list.</li></ul> <p>OR</p> <ul style="list-style-type: none"><li>explains why no changes to the code segment would be necessary.</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 lists are included in the List section of the Personalized Project Reference:<ul style="list-style-type: none"><li>Use the list(s) referenced in the written response to determine whether the point is earned.</li><li>If no list is referenced in the response, then use the first list to determine whether the point is earned.</li></ul></li><li>The response may earn this point if the described code modifications change the functionality of the program.</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 list (or other collection type) is not included in part (i) of the List section of the Personalized Project Reference.</li><li>The response does not apply to the list included in the List section of the Personalized Project Reference.</li><li>The use of the list is trivial and does not assist in fulfilling the program’s purpose.</li><li>The response describes a modification that makes the list unnecessary.</li><li>The response includes an explanation that is implausible, inaccurate, or inconsistent with the program.</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 Boolean expression in this selection is "(product == userAnswer)". In this expression, product is the correct product of multiplying the two numbers generated by the code, and userAnswer is what the user thinks the correct product of multiplying the two numbers is. Both product and userAnswer are integers. This expression will evaluate to true if product is equal to userAnswer. In other words, this expression will evaluate to true if what the user thinks the correct product of multiplying the two numbers is the same as what the correct product is. A specific set of values that will cause this expression to evaluate to true is if product equals to 12 and userAnswer also equals to 12. Since 12 is equal to 12, userAnswer is thus equal to product, (product ==userAnswer) evaluates to true, so the user got the question right.

2(b)

A modification another programmer could make that would cause this procedure to have a logic error is if they switched the first and second selection statements. The first selection statement is "if(product) == userAnswer" and the second selection statement is "else if(userAnswer <= product + 5 && userAnswer>=product-5)". If the order of these two were switched (and the else if on the second statement was switched to an if, while the if on the first statement was switched to an else if), when the user gets an answer correct, the answer would first go through "if(userAnswer <= product + 5 && userAnswer>=product-5)". This would evaluate to true, and the program would output "Close! The right answer was \*the answer\*.". However, even if the user got the answer right, since the first if statement is counted as true, the program will always give that answer, instead of acknowledging that the user was correct. That is because once the first if statement is deemed correct, the else if statements after that are ignored, including the selection statement that determines if the user's answer is equal to the true answer. Since as long as the user's answer is equal to the true answer, the user's answer is also within 5 of the true answer, the "if(userAnswer <= product + 5 && userAnswer>=product-5)" will always be activated first and cause the "else if(product) == userAnswer" to be ignored, which is a logic error because the code runs exactly how it is told to, but the true intention of the code is not met because the programmer made a mistake.

2(c)

There would be no necessary changes to the code segment. This is because the code segment in part(ii) of the List section uses the list and the length of the list, instead of any static numbers or terms. The code segment in part(ii)'s purpose is to generate two random numbers from the list above. To find the range of numbers it has to generate, it uses the list and length of the list, instead of a setnumber. The way that the program does it is that it considers the entire list and all of its elements, then chooses a random index from inside of that list out of all indices possible. Then the program takes the value of the element at that random index as the value for either number1 or number2. Thus, another programmer can add as many elements to the end of the list as they want, and the code segment in part(ii) will be unchanged,

because it actively takes both the list and the length of the list when the code is ran, instead of relying on any set values. Thus it will not need any changing of the length of the list or the contents of the list differ, or in this case any new elements are added at the end of the list.

2(a)

One boolean expression is "(a==0&& c==0)". The only way which this expression will evaluate to true is if the user inputs both "a" and "c" as the number zero. These values will make the expression evaluate to true because the first condition in the boolean, "a==0", is true. Since this is an AND statement, it takes both conditions to be true for the expression to evaluate as true. The value for "c" is 0, which makes the second condition true. Since both conditions being true, the boolean expression will evaluate to true.

2(b)

If a modification was made to the "for loop", and "i<33" is turned into "i>0", then it will create an infinite loop. The condition of the loop will always be true no matter what the user input is and will keep running forever. This will make the procedure get stuck on this loop, unable to run any other sections of the code. This loop will also continue to change the color of the pixels, and as the loop keeps going, the procedure will try to change the color of pixels that do not exist. Although the procedure is attempting to change the color of pixels that do not exist, the loop will continue to run forever.

2(c)

If another programmer added several new elements to the end of the list, this section of code would have to be modified because it is currently set to work if there are exactly 32 items in the list. If more items are added to the list, then the condition "x<33" would have to be changed. For example, if elements are added to the list so the list now contains 50 elements, the condition in the "for loop" will have to be changed to "x<51". This will make it so the loop runs 50 times and accesses each element in the list once.

2(a)

The first selection statement included in my program is `if((user_response>(len(profession_list)))or(user_response<=0))`. The boolean expression in the selection statement is `((user_response>(len(profession_list)))or(user_response<=0))`. The set of values that would cause the expression to evaluate to true is an integer inputted by a person using my program that is not between 1 and 10, for example 0 and 11. These values would cause the expression to evaluate to true because the boolean expression is looking for integers bigger than the numeric size of the list ,10, with the specific code segmet being `((user_response>(len(profession_list)))`, or any integer equal to or below zero,`(user_response<=0))`. So therefore, any integer not between 1 and 10 would cause the boolean expression to evaluate to true.

2(b)

The procedure is `def findCollegeForProfessionChoice(user_response)`. If a modder were to modify the code within `findCollegeForProfessionChoice(user_response)`, a modification the modder would have to make to cause a logic error is one that replaces the "or" in the first if staement of the procedure, if `((user_response>(len(profession_list)))or(user_response<=0))`, to an "and". The behavior of the procedure would change because before the modification the boolean expression only checked if one of the statements were true, either the inputted integer was below or equal to 0 or over 10, because it was an "or" expression, either `((user_response>(len(profession_list))) or (user_response<=0))`. But after modifying the expression to be an "and" statement, the boolean expression now checks both statements and because an integer cant both be below or equal to zero and over 10, a logic error would occur.

2(c)

The list in my program is `profession_list` . If another programmer were to add additional elements to the end of `profession_list`, no modifications would be needed for the for loop that uses `profession_list`, for key in `profession_dictionary`. This is the case for my program because the only use for `profession_list` in for key in `pfession_dictionary` is to line the selected value from a person using my code in `profesion_list` to the identical key in `profession_dictionary` which prints out the information related to that key, for example if the selected element from `profession_list` is education that value would be aligned to education the key in `profession_dictionary` and print out the key and its information related to it like the colleges recomended for a major in education. The only modification that would be neccassary after adding elements to `profession_list` is putting identical keys of the newly added elements from `profession_list` into `profession_dictionary` and giving information related to the key so that if any of the newly added elements in `profession_list` are slected from a person using my code, the code will still function because of the new keys and information in `profesion_dictionary` and not have an error.

2(a)

The first selection statement in my custom procedure, is the "repeat until (slot 2  $\neq$  slot 1)" block. Ultimately, this generates random countries from the desired flag set until slot 2 is not equal to slot 1. Under the circumstance that "flag set" and "word set" represent the easy flags, and "question # = 1", a generated value that would work is "USA". This is because, slot 2 and slot 1 placeholder the values of 2 of the 4 multiple choice options. Slot 1 has been defined to "item (question #) of (word set)", and Slot 2 has been set to any random item from the easy flags set. So, slot 1 would be defined as China, and slot 2 could generate the USA. Since USA  $\neq$  China, this boolean statement would return true, so it wouldn't have to regenerate a new option. However, if slot 2 had been defined as "China", then the boolean statement would yield false, requiring it to regenerate slot 2 until the boolean yields true.

2(b)

If another programmer changed one of my boolean statements, my project could face severe logic errors as it will either stop giving valid answer choices or it has a chance to give duplicate answer choices. The most complex boolean statement, would be the third one, which has 2 conditions needed to be met. However, if the second condition, "and (slot 1)  $\neq$  (slot 4)", is deleted, there will always be one answer choice that may give duplicate options. This is because, instead of "slot 4" checking with all 3 other "slots" for duplicate options, it only checks with "slots" 2 and 3. So, although the program functions properly, because it doesn't check for duplicate options with "slot 1", there is a higher chance of getting duplicate right answers. This would lead to it producing answer choices like: "China, France, India, China", or even "Australia, Australia, India, Canada" after it is put through a randomizer. This doesn't only happen for China and Australia, it can happen for all of them. So, this could leave confusion for the viewers as invalid, duplicate options have been produced. Although this change may seem miniscule to the other programmer, this could lead to logic error through the creation of invalid answer choices. Although they could argue that the program still functions, it is useless if the answer choices do not make sense.

2(c)

Taking the list, "Easy flags (words)", into account, if an **n** number of element were added to the end of the list, it could work in 2 different ways depending on the intention of my code, assuming the elements are countries. I could either leave it the same way, or change it. The first way, would be to leave it exactly the same, because the way my code has been written, nothing will be affected except for the fact that there will be a greater variety of answer choices. This is because, my program will only ask questions for items 1-10 of "Easy flags (words)". So, the only portion where the excess elements will be incorporated is during the custom procedure "Answer selection ( ) ( ) ( )". This will only be found in the answer choices, as my code says to take a random item from the "word set". So, it won't affect the questions being asked to users at all. However, the second way, is to modify the number of questions asked. I would need to change my "for loop" from "for (i) = (1) to (10)" to "for (i) = (1) to (10+n)". This simply increases the number of questions being asked, however along with the

addition of questions, I must add an ***n*** number of flags to correspond with the question, otherwise it will prompt the flag of Russia, an extra ***n*** number of times. However, if the elements added to the list weren't countries at all, I would simply have to modify the "word set" being put into my custom procedure. I would simply have to add an extra block before the "for loop" to create a new variable called "10 easy flags" which would be set to equal "items (1 to 10) of (Easy flags (words))" Then, I would take this newly updated list and drag it into the third parameter of my custom procedure, to ensure that all of the words being used are actual countries. My code is able to have a variety of methods to handle this situation, due to how the custom procedure manages complexity. To use my custom procedure efficiently, I added extra parameters to set what the "word set" and "flag set" is equal to in the beginning itself, so that I don't have to keep dragging "easy flags (words)". Due to these parameters, I am able to minimally modify my procedures, while averting the problem completely, with choices depending on my intentions. If they elements were countries I could either have a variety of options, or an increased question base. Due to these parameters, I am able to easily, efficiently, work around this situation and modify and fix the code to work again. In addition, the reason my procedure is able to work around this so easily is because the way it was coded allows for changes to be made in different locations, addressing different results for the program, utilizing redundancy through multiple ways of fixing problems in case one method doesn't work.

2(a)

The first selection statement included in the procedure section of my program is `var amount == number (getText( "amountInput"))`; the boolean expression is a OR statment. If the procedure is false it means the user inputed the correct value.

category is not left open

amount is a number

amount is greater than 0

2(b)

If an other programer modifies my program and couse it have a logic error

it will be the loop part. If the programer changed the `i=0:` to `i= 1`. This will cause program to be fromated different. But the itrater of the expanse in the expanseslist and then finds `totalexpanse` and the reaming balance will be the same.

2(c)

The list included in my program is `expensesList`. If a programer addes a some new element it would not change any thing. If the element is expances then procedure will just add it up to find the `totalexperiences` .After that the program subrates the `totalexperiences` from the income to find the reamingbalance

2(a)

In the procedure the value that allows the boolean to be true is variable "i". It checks through the values of the index. "i" allows the boolean to be true by taking the value of the "i" in the index of count scoreboard and comparing it to the top score. "i" will only display true to numbers inclusive 1-5 due to the index of Count Score Board only holding 5 values in its index.

2(b)

The way that a user would could modify the code that would result in a logic error, is by trying to set "i" higher than 5. That will cause the code not to run. That will cause there to be a skip on the first 5 values of the index, leading to a logic error.(There are only 5 values in the Count Score Board index).

2(c)

If new elements were to be added to the list it would allow the previous modification to check the additional values of the list "i" was to compare to variable top score. To correct these changes simply set "i" back to 1 and it will check all all the values of the index with no problem. Then allow the procedure to continue comparing the top score of count score board.

## Question 2

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

### Overview

**NEW for 2025:** The question overviews can be found in the *Chief Reader Report on Student Responses on AP Central*.

### Sample: A

#### Score:

**Question 2(a): 1**

**Question 2(b): 1**

**Question 2(c): 1**

#### Question 2(a):

The response earned this point, demonstrating all three criteria:

- The response identifies “(product == userAnswer)” as the Boolean expression in the first selection statement.
- The response identifies a specific set of values that will cause that will cause the Boolean expression to evaluate to `true`: “A specific set of values that will cause this expression to evaluate to true is if product equals to 12 and userAnswer also equals to 12.”
- The response explains why the specified set of values will cause the expression to evaluate to `true`: “since 12 is equal to 12, userAnswer is thus equal to product, (product ==userAnswer) evaluates to true, so the user got the question right.”

#### Question 2(b):

The response earned this point, demonstrating both criteria:

- The response describes “if they switched the first and second selection statements” as a modification that will cause the procedure to have a logic error.
- The response describes how the behavior of the procedure would change: “If the order of these two were switched...when the user gets an answer correct...the program would output “Close! The right answer was \*the answer\*.”. However, even if the user got the answer right...the program will always give that answer, instead of acknowledging that the user was correct.”

#### Question 2(c):

The response earned this point.

The response explains why no changes to the code segment would be necessary if another programmer adds several new elements to the end of the list: “This is because the code segment in part(ii) of the List section uses the list and the length of the list, instead of any static numbers or terms...it uses the list and length of the list, instead of a setnumber. The way that the program does it is that it considers the entire list and all of its elements, then chooses a random index from inside of that list out of all indices possible.”

**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, demonstrating all three criteria:

- The response identifies “(a==0&& c==0)” as the Boolean expression.
- The response identifies a specific set of values that will cause the Boolean expression to evaluate to `true`: “the user inputs both “a” and “c” as the number zero.”
- The response explains why this set of values will cause the expression to evaluate to `true`: “These values will make the expression evaluate to true because the first condition in the boolean, “a==0” , is true. Since this is an AND statement, it takes both conditions to be true for the expression to evaluate as true. The value for “c” is 0, which makes the second condition true. Since both conditions being true, the boolean expression will evaluate to true.”

Question 2(b):

The response earned this point, demonstrating both criteria:

- The response describes a modification that will cause the procedure to have a logic error: “If a modification was made to the for loop and `i<33` is turned into `i>0`”.
- The response describes how the behavior of the procedure would change: “then it will create an infinite loop”.

Question 2(c):

The response earned this point.

The response explains how the code segment would need to change if another programmer adds several new elements to the end of the list: “the condition “`x<33`” would have to be changed. For example, if elements are added to the list so the list now contains 50 elements, the condition in the “for loop” will have to be changed to “`x<51`”.”

**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, demonstrating all three of the criteria:

- The response identifies “`((user_response > len(profession_list))) or (user_response <= 0)`” as the Boolean expression in the first selection statement.
- The response identifies “an integer inputted by a person using my program that is not between 1 and 10, for example 0 and 11” as values that will cause the Boolean expression to evaluate to `true`. The response explains why the specified set of values will cause the expression to evaluate to `true`: “These values would cause the expression to evaluate to true because the boolean expression is looking for integers bigger than the numeric size of the list, 10, with the specific code segment being `((user_response > len(profession_list)))`, or any integer equal to or below zero, `(user_response <= 0)`. So therefore, any integer not between 1 and 10 would cause the boolean expression to evaluate to true.”

Question 2(b):

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

- The response describes “one that replaces the “or” in the first if statement of the procedure, if `((user_response > len(profession_list))) or (user_response <= 0)`, to an “and”” as a modification that will cause the procedure to have a logic error.
- The response states, “the boolean expression now checks both statements and because an integer can't both be below or equal to zero and over 10, a logic error would occur,” but does not describe how the behavior of the procedure would change.

Question 2(c):

The response earned this point.

The response explains how the code segment would need to change if another programmer adds several new elements to the end of the list: “The only modification that would be necessary after adding elements to `profession_list` is putting identical keys of the newly added elements from `profession_list` into `profession_dictionary` and giving information related to the key so that if any of the newly added elements in `profession_list` are selected from a person using my code, the code will still function because of the new keys and information in `profession_dictionary` and not have an error.”

## Question 2 (continued)

**Sample: D**

**Score:**

**Question 2(a): 0**

**Question 2(b): 1**

**Question 2(c): 1**

Question 2(a):

The response did not earn this point, demonstrating two of the three criteria:

- The response does not identify the Boolean expression in the first selection statement. The response identifies “repeat until (slot 2  $\neq$  slot 1),” which is an iteration statement. The response does not include a selection statement.
- The response identifies “slot 1 would be defined as China, and slot 2 could generate the USA” as the set of values that will cause the Boolean expression to evaluate to `true`. The response explains why the specified set of values will cause the expression to evaluate to `true`: “Since `USA  $\neq$  China`, this boolean statement would return true.”

Question 2(b):

The response earned this point, demonstrating both criteria:

- The response describes “if the second condition, ‘and (slot 1)  $\neq$  (slot 4)’, is deleted” as a modification that will cause the procedure to have a logic error.
- The response describes how the behavior of the procedure would change: “This would lead to it producing answer choices like: “China, France, India, China”, or even “Australia, Australia, India, Canada”” and “So, although the program functions properly, because it doesn’t check for duplicate options with “slot 1”, there is a higher chance of getting duplicate right answers.”

Question 2(c):

The response earned this point.

The response explains that the program will work either way. The response explains how the code segment would need to change if another programmer adds several new elements to the end of the list: “if an ***n*** number of element were added to the end of the list...is to modify the number of questions asked. I would need to change my “for loop” from “for (i) = (1) to (10)” to “for (i) = (1) to (10+***n***)”” The response also explains why no changes to the code segment would be necessary: “the way my code has been written, nothing will be affected except for the fact that there will be a greater variety of answer choices. This is because, my program will only ask questions for items 1-10 of “Easy flags (words) “.”

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

Question 2(a):

The response did not earn this point, demonstrating none of the three criteria:

- The response does not identify the Boolean expression in the first selection statement. Although the response states “the boolean expression is a OR statment,” the response does not specify the details of this logical expression. Additionally, the response incorrectly identifies the selection statement: “The first selection statement included in the procedure section of my program is `var amount == number (getText( “amountInput”))`.”
- The response identifies “category is not left open,” “amount is a number,” and “amount is greater than 0,” but this set of values will not cause the expression to evaluate to `true`. The response does not explain why the specified set of values will cause the expression to evaluate to `true`.

Question 2(b):

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

- The response describes how changing the initialization of `i` in the for loop to 1 will cause the procedure to have a logic error.
- The response describes how the behavior of the procedure would not change by stating that total expenses will remain the same, but total expenses would change since the first element of the expense list will not be included in the sum.

Question 2(c):

The response earned this point.

The response explains why no changes to the code segment would be necessary if another programmer adds several new elements to the end of the list: “If the element is expances then procedure will just add it up to find the totalexpenses.”

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

Question 2(a):

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

- The response does not identify the Boolean expression in the first selection statement.
- The response identifies “numbers inclusive 1-5” as the set of values that will cause the Boolean expression to evaluate to `true`.
- The responses does not explain why the specified set of values will cause the expression to evaluate to `true`: “due to the index of Count Score Board only holding 5 values in its index.” This explains why these are value index values, not why these values make the Boolean expression `true`.

Question 2(b):

The response earned this point, demonstrating both criteria:

- The response describes “set “i” higher than 5” as a modification that will cause the procedure to have a logic error.
- The response describes how the behavior of the procedure would change: “That will cause there to be a skip on the first 5 values of the index.”

Question 2(c):

The response did not earn this point.

The response explains how the code segment would need to change if another programmer adds several new elements to the end of the list: “If new elements were to be added to the list it would allow the previous modification to check the additional values of the list “i” was to compare to variable top score. To correct these changes simply set “i” back to 1.” This explanation is not correct because the response is correcting the error introduced in 2(b), not handling the addition of new elements to the list.