
AP[®] 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
Written Response 2a: Algorithm Development (0–1 points)	<p>The written response:</p> <ul style="list-style-type: none"> • identifies the Boolean expression in the first selection statement. • identifies a specific value or set of values that will cause the Boolean expression of the selection statement to evaluate to <code>false</code>. • explains why the specified value or set of values will cause the expression to evaluate to <code>false</code>. 	<p>Consider the Personalized Project Reference and Written Response 2(a) when scoring this point.</p> <ul style="list-style-type: none"> • 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. • The selection statement does not need to be contained in a procedure to earn this point. • The response does not have to explicitly state the Boolean expression as long as it is described. • The identified Boolean expression can include the header of the selection statement (e.g., <code>if (x > 2)</code> instead of <code>x > 2</code>). • The response may earn this point for a selection statement that either does or does not contain an <code>else</code> clause. <p>Do NOT award a point if any one or more of the following is true:</p> <ul style="list-style-type: none"> • The Procedure section of the Personalized Project Reference does not contain a selection statement. • The identified Boolean expression does not match the code in the first selection statement. • The identified value or set of values that will cause the expression to evaluate to <code>false</code> does not match the code in the first selection statement. • The response identifies a Boolean expression or value (or set of values) that is implausible, inaccurate, or inconsistent with the program.

Reporting Category	Scoring Criteria	Decision Rules
Written Response 2b: Errors and Testing (0–1 points)	<p>The written response:</p> <ul style="list-style-type: none"> describes a modification another programmer could make to the code that would result in a logic error. explains why this modification would result in a logic error. 	<p>Consider the Personalized Project Reference and Written Response 2(b) when scoring this point.</p> <ul style="list-style-type: none"> The modification can be changing code, adding code, and/or deleting code. The modification does not need to be to the list itself; it can be to any part of the code included in part (ii) of the List section of the Personalized Project Reference. 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. The response may describe more than one modification that leads to a logic error. If the response describes more than one modification, every modification must be explained correctly. <p>Do NOT award a point if any one or more of the following is true:</p> <ul style="list-style-type: none"> A <i>list</i> (or other <i>collection type</i>) is not included in part (ii) of the List section of the Personalized Project Reference. The response does not apply to the code in part (ii) of the List section of the Personalized Project Reference. The response states only that the program will not run without further explanation. The response describes a modification or change in behavior that is implausible, inaccurate, or inconsistent with the program. The response includes an explanation that is implausible, inaccurate, or inconsistent with the program or the described modification.
Written Response 2c: Data and Procedural Abstraction (0–1 points)	<p>The written response:</p> <ul style="list-style-type: none"> describes the functionality provided by the identified procedure. explains how implementing this functionality as a procedure results in the program being easier to maintain than if the functionality were not implemented as a procedure. 	<p>Consider the Personalized Project Reference and Written Response 2(c) when scoring this point.</p> <ul style="list-style-type: none"> If multiple procedures are included in part (i) of the Procedure section of the Personalized Project Reference: <ul style="list-style-type: none"> Use the procedure referenced in the written response to determine whether the point is earned. If no procedure is referenced in the written response, then use the first procedure to determine whether the point is earned. The written response does not need to reference a parameter; however, if it does reference a parameter, the parameter can be explicit or implicit. <p>Do NOT award a point if any one or more of the following is true:</p> <ul style="list-style-type: none"> A procedure is not included in part (i) of the Procedure section of the Personalized Project Reference. The response does not apply to the procedure in part (i) of the Procedure section of the Personalized Project Reference. The response includes an explanation that is implausible, inaccurate, or inconsistent with the procedure.

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 my selection statement is "if item i of *manipulator* = '(space value)' then:". A set of values that would cause this expression to evaluate to false is any character that is not a space value. These values causes this expression to be false because they are not equivalent to a space value or " ".

2(b)

A modification that another programmer could make to this code segment is to change the "repeat length of *manipulator*" to "repeat until $i > (\text{length of } \textit{manipulator}) + 1$ ". This would result in attempting to index an item of the list which would not exist, thus creating a logic error.

2(c)

The functionality provided by "*takeoutspaces*" is removing the spaces in a string. This procedure makes my program easier to maintain because it is used multiple times with various inputs, which would result in a lot more messy code if it wasn't implemented as a procedure. The procedure manages complexity by being abstract, which means if someone wanted to modify the code in the future, they could do so without having to alter many different pieces of code.

2(a)

The boolean expression in this selection statement is comparing whether or not `pygame.time.get_ticks()` is greater than or equal to `next_tree_time`. A specific set of values that will cause this expression to evaluate to false is any time the assigned `next_tree_time` is greater than `pygame.time.get_ticks()`. For example, if `next_tree_time = 1200` and `pygame.time.get_ticks() = 800`, 800 is not greater than or equal to 1200 so the expression will evaluate to false.

2(b)

A modification that another programmer could make to this code segment that would result in a logic error is using `+= y` instead of `-= y`. This is because adding `y` to the center of a tree in the list will make the tree go down rather than go up. This will lead to no trees ever appearing on screen since due to the fact that they spawn from the bottom, they will first spawn off screen and then go down so the user never sees them. This ultimately breaks the entire game as the player can't lose if no trees are going upward which ruins the intended purpose of the program through this very simple change in the code that results in this logic error.

2(c)

The functionality of this procedure is that it first goes through all of the trees in `tree_list` and moves them up and then draws them again on the screen in the correct position. In addition, the procedure checks to see if `pygame.time.get_ticks()` is greater than or equal to `next_tree_time` and if it is, a new tree is created with a random horizontal position across the screen area and is then appended to the tree list, and after a new `next_tree_time` is set based on the current time plus a random interval and it is returned. However, if `next_tree_time` is greater than `pygame.time.get_ticks()`, the same `next_tree_time` is returned that was put in as the parameter. Implementing this functionality as a procedure results in the program being easier to maintain than if the functionality were not implemented as a procedure because as a result of being a procedure, we know that this code should be doing its intended purpose and that we can abstract the details of how the procedure actually functions out of the main program. This allows for much easier maintenance later when you might have to tweak parts of the program that aren't a procedure while our procedure can remain untouched and abstract unless absolutely necessary.

2(a)

The Boolean expression in the first selection statement is "if temp_today>hot_temps[i]" on line 12. An example of values that would cause the expression to evaluate to "false" is the value of "temp_today" being assigned the number 65 and the value of "hot_temps[i]" being assigned the number 100. This will cause this expression to evaluate to "false" because 65 is not larger than 100.

2(b)

A modification another programmer could make to this code that would result in a logic error is changing the boolean expression "if temp_today>hot_temps[i]" on line 12 to "if temp_today/0=1" as you cannot divide by zero. Dividing a number by zero is not possible logically and will result in the program stopping and reporting the error "divide by zero".

2(c)

The functionality provided by this procedure is for people who want to compare the temperature today (most likely the temperature that they are experiencing) to the hottest and coldest temperatures in various different countries by looking at the difference between the temperature inputted by the user of the program and the hottest and coldest temperatures various countries. Implimenting this functionality as a procedure is easier to maintain than if the functionality were not implimented as a procedure because this way the person who wants to make these temperature comparisons doesn't have to calculate the difference by hand for each countries' hottest and coldest temperatures and can just input the temperature they want to compare with each countries' hottest and coldest temperatures. The program makes these comparisons less time consuming and mentally taxing and therefore easier to maintain.

2(a)

In my Personalized Project Reference a Boolean expression included is whether the "length of 'mood log' is greater than 7". Values that will cause the expression to evaluate to false would include the length of the list being 7 or less. This is because in order for the code of segment that clears the list to operate, the list must be greater than 7, meaning there is a start to a new week of mood tracking. This is included in the procedure because the program is meant to store a user's moods in a list for a week. So I added this Boolean expression in order to erase the list if there is more than several moods logged for the week and this segment of my code is repeated everytime the user wants to track their mood, so that way the length of the list is always being checked.

2(b)

In the code segment, a modification made by another programmer that would result in a logic error could be editing "happy" from the "mood user is in = happy". This part of the code is meant to find the affirmations in the "words of affirmation" list that match the user's mood that they logged. SO when a user inputs they are happy, this code segments is used so that the sprite on the user's screen says a affirmation that congratulates them on their happy mood, compared to an affirmation made for a sad person which would have told the user words of encouragement. If a different programmer were to change this part of the code to one example being "sad" it could lead to a user entering 'happy' as their mood, and receiving an affirmation which is meant to be catered towards a sad user. By editing that part of the code segment, the resulting logic error due to the modification could confuse users.

2(c)

The procedure in my program is a way to organize the inputs of the user and manage it. The user answers two questions and inputs two different inputs, one being the mood they are track, and the other being the day they are tracking the mood for. The procedure was built in order to track these inputs into an embedded list within "mood logs". This made the code segment including this procedure to run effectively because it allows the program to ask the user for two different inputs but store this inputs as one index. So by implementing this procedure, it allows the rest of my program to easily analyze the inputs from the user. In my program, the list which stores the user's mood and the day they tracked the mood for is used in order to draw faces on a chart where the faces match the users mood, and the placement of the faces are dependent on what day the user inputs their mood. So by adding this it allows code segments to analyze and search each input made by the user easily and more efficiently. If I didn't add this the program would have a harder time tracking the different parts of the user's inputs, and using that data as required..

2(a)

I believe I submitted an incorrect image as I meant to submit a previous function which included several booleans like:

```
if letter[i].includes letter
setProperty letter[i] "background-color", "green"
else if answer.includes letter
setProperty letter[i], "background-color", "green"
else
setProperty letter[i], "background-color", "gray"
```

(Not exactly like this but very similar since I cannot remember the exact variable names and what this included without the picture)

I will be discussing this previous section instead of the one I submitted.

The boolean expression will evaluate to false when the letter which is submitted by the user evaluates to false rather than true. When the letters are not the same, they do not equal each other, thus setting the value to false instead of true and moving on to check if the value is yellow. Any value in which the input letter and correct letter do not match, will cause the program to be set to false.

2(b)

If another programmer wanted to add another row to the list of words, they might correctly add the row to the "words" variable and correctly define where the row is using getColumn but neglect to match the new row to the length of the previous rows (each row needs to be the same length), then there might be a logic error when a new word is called.

When the program runs, a random word that is the length of row1 (the same as the other rows) is chosen from one of the 4 rows at random. If someone added a 5th row at a shorter length, there may be no word at the location where the program calls for one, resulting in an error/result that is not a word. For example, if a random index number of 55 was made (index variable) and row5 was randomly selected (words variable) and row5 is only 32 words in length, then there would be no word available at row5 for index = 55.

2(c)

The procedure in the 4th line (the for statement) makes the process of resetting the background colors and letters much more efficient. By using a procedure, every square is reset from green/yellow/gray and cleared of any letters in 3 lines of code rather than the alternative of writing 60 lines of code that would do the same thing. Had I not used a procedure, errors would be much more likely and it would be extremely time consuming. I can also reuse this easily if I wanted to add/remove text boxes. When creating this project, I had to add another row of boxes (5 boxes) and it was very easy/efficient to simply change the value of $i < 25$ to $i < 30$ and get the program working properly.

2(a)

My boolean expression is in my function, in part i of my procedure. The boolean compares the string `user_answers` with the `i`-th term of `correct_answers`. If `user_answers` perfectly matches the `i`-th term of `correct_answers` then `user_answer == 0` indicating it to be true. When the function returns 0 the expression evaluates to false. In the same way the function checks for right answers it checks for wrong ones as well. Using string compare the expression compares `user_answers` to `correct_answers` if they don't perfectly match then the expression returns 0 indicating false.

2(b)

A modification error made by the other programmer than would result in logic error would be changing the plus sign to a minus sign on the score calculation part of part(ii). Changing the `score+=` to a `score -=` would subtract one from the score which would still allow the code to run but result in an overall score of 0 no matter if the user got any questions right. The user would still be able to answer all questions but would not be able to see their score.

2(c)

The procedure `check answers` takes a call such as `checkanswers("LeBron James", correct_answers[0], 4)` and compares the user-answer "Lebron James" with `correct_answers` array. If the user-answer matches with `correct_answers` then the function returns a value of 1, adding 1 to the score. Implementing the function as a procedure allows the function to be called in multiple areas and having a singular function rather than there being multiple functions across my code. Having a singular procedure also makes it easier to change, so if there are problems it is easier to identify or if I desired a different functionality.

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 “if item i of manipulator = ‘(space value)’ then:” as the Boolean expression in the first selection statement. The response includes both the “if” keyword and the Boolean expression, which is acceptable to meet this criterion.
- The response identifies “any character that is not a space value” as the set of values that will cause the Boolean expression to evaluate to `false`.
- The response explains why the set of values will cause the expression to evaluate to `false`: “These values causes this expressions to be false because they are not equivalent to a space value or “.”

Question 2(b):

The response earned this point, demonstrating both criteria:

- The response describes “change the “repeat length of manipulator” to “repeat until i > (length of manipulator) + 1”” as a modification that will cause the code segment to have a logic error.
- The response describes why this modification would result in a logic error: “This would result in attempting to index an item of the list which would not exist, thus creating a logic error.”

Question 2(c):

The response earned this point, demonstrating both criteria:

- The response describes the functionality provided by the identified procedure: “The functionality provided by “takeoutspaces” is removing the spaces in a string.”
- The response explains how implementing this functionality as a procedure results in the program being easier to maintain: “This procedure makes my program easier to maintain because it is used multiple times with various inputs, which would result in a lot more messy code if it wasn’t implemented as a procedure.”

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 “comparing whether or not `pygame.time.get_ticks()` is greater than or equal to `next_tree_time`” as the Boolean expression in the first selection statement.
- The response identifies “`next_tree_time = 1200` and `pygame.time.ticks() = 800`” as the set of values that will cause the Boolean expression to evaluate to `false`.
- The response explains why the set of values will cause the expression to evaluate to `false`: “800 is not greater than or equal to 1200 so the expression will evaluate to false.”

Question 2(b):

The response earned this point, demonstrating both criteria:

- The response describes “using `+= y` instead of `-= y`” as a modification that will cause the code segment to have a logic error.
- The response describes why this modification would result in a logic error: “This is because adding `y` to the center of a tree in the list will make the tree go down rather than go up. This will led to no trees ever appearing on screen since due to the fact that they spawn from the bottom, they will first spawn off screen and then go down so the user never see’s them.”

Question 2(c):

The response earned this point, demonstrating both criteria:

- The response describes the functionality provided by the identified procedure: “The functionality of this procedure is that it first go’s through all of the trees in `tree_list` and moves them up and then draw them again on the screen in the correct position...a new tree is created with a random horizontal position across the screen area and is then appended to the tree list.”
- The response explains how implementing this functionality as a procedure results in the program being easier to maintain: “Implementing this...results in the program being eaiser to maintain... because as a result of being a procedure...we can abstract the details of how the procedure actually functions out of the main program. This allows for much eaiser maintanince later when you might have to tweak parts of the program that aren’t a procedure while our procedure can remain untouched and abstract unless absolutly necessary.”

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

Question 2(a):

The response earned this point, demonstrating all three criteria:

- The response identifies “if temp_today>hot_temps[i]” as the Boolean expression in the first selection statement. The response includes both the “if” keyword and the Boolean expression, which is acceptable to meet this criterion.
- The response identifies “number 65 and the value of “hot_temps[i]” being assigned the number 100” as the values that will cause the Boolean expression to evaluate to `false`.
- The response explains why the values will cause the expression to evaluate to `false`: “This will cause this expression to evaluate to “false” because 65 is not larger than 100.”

Question 2(b):

The response earned this point, demonstrating both criteria:

- The response describes ““if temp_today>hot_temps[i]” on line 12 to “if temp_today/0=1”” as a modification that will cause the code segment to have a logic error.
- The response describes why this modification would result in a logic error: “you cannot divide by zero.”

Question 2(c):

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

- The response describes the functionality provided by the identified procedure: “The functionality provided by this procedure is for people who want to compare the temperature today.”
- The response does not explain how implementing this functionality as a procedure results in the program being easier to maintain. The response discusses how implementing this function benefits the user.

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, demonstrating all three criteria:

- The response identifies “length of ‘mood log’ is greater than 7” as the Boolean expression in the first selection statement.
- The response identifies “the length of the list being 7 or less” as the set of values that will cause the Boolean expression to evaluate to `false`.
- The response explains why the set of values will cause the expression to evaluate to `false` by explaining when the expression will evaluate to `true`: “in order for the code of segment that clears the list to operate, the list must be greater than 7.”

Question 2(b):

The response earned this point, demonstrating both criteria:

- The response describes “editing “happy” from the “mood user is in = happy”” as a modification that will cause the code segment to have a logic error.
- The response describes why this modification would result in a logic error: “If a different programmer were to change this part of the code to one example being “sad” it could lead to a user entering ‘happy’ as their mood, and receiving an affirmation which is meant to be catered towards a sad user.”

Question 2(c):

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

- The response describes the functionality provided by the identified procedure: “The user answers two questions and inputs two different inputs, one being the mood they are track, and the other being the day they are tracking the mood for. The procedure was built in order to track these inputs into an embedded list within “mood logs”.”
- The response does not explain how implementing this functionality as a procedure results in the program being easier to maintain. The response states: “If I didn’t add this the program would have a harder time tracking the different parts of the user’s inputs, and using that data as required.” This does not explain why the program is easier to maintain.

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

Question 2(a):

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

- The response does not identify the Boolean expression in the first selection statement. The response incorrectly identifies a selection statement as a Boolean expression. There is no selection statement in the Procedure section of the Personalized Project Reference.
- The response does not identify a specific value or set of values that will cause the Boolean expression to evaluate to `false`: “when the letter which is submitted by the user evaluates to false rather than true.” However, this describes the value of the Boolean expression, not the value that causes the Boolean expression to be `false`.
- The response explains why the value will cause the expression to evaluate to `false`: “When the letters are not the same, they do not equal each other, thus setting the value to false instead of true.”

Question 2(b):

The response earned this point, demonstrating both criteria:

- The response describes a modification that will cause the code segment to have a logic error: “they might correctly add the row to the “words” variable and correctly define where the row is using `getColumn` but neglect to match the new row to the length of the previous rows.”
- The response describes why this modification would result in a logic error: “When the program runs, a random word that is the length of `row1` (the same as the other rows) is chosen from one of the 4 rows at random. If someone added a 5th row at a shorter length, there may be no word at the location where the program calls for one, resulting in an error/result that is not a word.”

Question 2(c):

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

- The response does not describe the functionality provided by the identified procedure. The response refers to the procedure as “the 4th line (the for statement)” and then describes the functionality of the for-loop: “the process of resetting the background colors and letters.”
- The response does not explain how implementing this functionality as a procedure results in the program being easier to maintain. Instead, the response explains how the for-loop specifically makes the procedure easier to maintain by shortening the code and making it easy to change the number of rows of boxes.

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

Question 2(a):

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

- The response identifies “The boolean compares the string `user_answers` with the *i*-th term of correct answers” as the Boolean expression in the first selection statement.
- The response does not identify a value that will cause the Boolean expression to evaluate to `false`. The response states, “When the function returns 0 the expression evaluates to false,” but the response does not provide a specific value to lead to this outcome.
- The response does not explain why a specific value leads to the Boolean expression evaluating to `false` since no specific value is identified.

Question 2(b):

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

- The response describes “Changing the `score+=` to a `score -=`” as a modification that will cause the code segment to have a logic error.
- The response describes why this modification would result in a logic error. The response states that the error will “result in an overall score of 0 no matter if the user got any questions right.” However, this is incorrect. The result will be that the score will be the negative score of how many questions the user gets right.

Question 2(c):

The response earned this point, demonstrating both criteria:

- The response describes the functionality provided by the identified procedure: “compares the user-answer “Lebron James” with `correct_answers` array. If the user-answer matches with `correct_answers` then the function returns a value of 1, adding 1 to the score.”
- The response explains how implementing this functionality as a procedure results in the program being easier to maintain: “allows the function to be called in multiple areas and having a singular function rather than there being multiple functions across my code.”