# Chief Reader Report on Student Responses:

## 2023 AP® Computer Science A Free-Response Questions

| | | | | |
|---|---|---|---|---|
| • Number of Students Scored | 94,438 | | | |
| • Number of Readers | 483 | | | |
| • Score Distribution | | Exam Score | N | %At |
| | | 5 | 25,269 | 26.76 |
| | | 4 | 21,181 | 22.43 |
| | | 3 | 17,730 | 18.77 |
| | | 2 | 8,984 | 9.51 |
| | | 1 | 21,274 | 22.53 |
| • Global Mean | 3.21 | | | |

The following comments on the 2023 free-response questions for AP® Computer Science A were written by the Chief Reader, Don Blaheta, Associate Professor of Computer Science at Longwood University. They give an overview of each free-response question and of how students performed on the question, including typical student errors. General comments regarding the skills and content that students frequently have the most problems with are included. Some suggestions for improving student preparation in these areas are also provided. Teachers are encouraged to attend a College Board workshop to learn strategies for improving student performance in specific areas.

## Question 1

**Task:** Methods and Control
**Topic:** Appointment Book
**Max Score:** 9
**Mean Score:** 4.07

***What were the responses to this question expected to demonstrate?***

This question tested the student's ability to:

- Write program code to create objects of a class and call methods.
- Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

More specifically, this question assessed the ability to iterate through a range, determine when a proper stopping condition has been met before the end of the range, call instance methods from other instance methods of the same class, and use a method's return value in a conditional expression.

In part (a) students were asked to find the first available block of a specified duration within a specified hour, using the class (static) method `isMinuteFree` to determine availability. To do this, they had to initialize a variable to track whether a block of the requested length could fit in the block of available minutes starting at each candidate starting minute. They had to iterate through all necessary minutes in the hour, call the given method on each necessary minute, and conditionally update the tracking variable—either to continue tracking the candidate block or to reset for the next candidate block, as appropriate. After the loop, they had to return `-1` if no block of `duration` or more minutes was available. There were essentially two interacting algorithms in part (a): keeping track of contiguous blocks of free minutes and choosing the correct starting minute (or `-1`).

In part (b) students were asked to reserve the first available block of a specified duration within a specified set of hours. They were instructed to use the method written in part (a), `findFreeBlock`, to determine the appropriate minute to book and to use another method from the class's API, `reserveBlock`, to book the appointment, then return `true`. They had to demonstrate the ability to determine if a reservation could *not* be made, in which case the method needed to return `false` after the loop.

***How well did the responses address the course content related to this question? How well did the responses integrate the skills required on this question?***

*Write program code to create objects of a class and call methods.*

Both parts of this question involved calling instance methods from another method of the same class. In part (a) responses needed to call `isMinuteFree` with the parameter `period` and an integer representing a minute; in part (b) responses needed to call the `findFreeBlock` method from part (a) with a period and the parameter `duration` and call `reserveBlock` with a period, a minute, and the duration.

Although most responses called the methods syntactically correctly, there were some common errors. Some responses confused method declaration and method call syntax by adding extraneous types to the arguments in the method call. Some responses invoked the method using the class name despite it being an instance method, not a class (static) method. Many responses that chose to re-call `findFreeBlock` for the second argument to `reserveBlock` became confused by the nested method call and ended up omitting the third parameter to `reserveBlock`.

*Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.*

Both parts of this question involved iterating over a range and using conditional logic to control the operation at each step of the iteration.

In part (a) responses had to iterate to identify the earliest block of contiguous free minutes of a given duration. Responses typically used an appropriate state variable—an integer counter of the length of an accumulating block or a Boolean status indicating whether the accumulating block remained contiguous—and initialized it correctly before use. Many responses did not cover the required range correctly, either because they considered one too few minutes due to a bounds error or because they considered minutes beyond the end of the range due to missing or incorrect logic in a nested loop solution. Most responses had an appropriate comparison with `duration`; however, many responses had an algorithmic sequencing error in which a successful return of the starting minute came too late (resulting in booking a later appointment than the first available one or booking multiple appointments) or an unsuccessful return of `-1` came too early (before all necessary minutes had been considered). Another common algorithmic error was checking only the two endpoints of a potential block, rather than all the contiguous minutes within the block.

In part (b) responses had to iterate over a specified subset of the periods in a day to find and reserve the earliest available block of the specified duration. Most responses correctly bounded their iteration to include the starting and ending periods. Most responses called `findFreeBlock` correctly and many responses attempted to use the result of a call to `findFreeBlock` as a guard. Most responses did not store the result in a variable and instead made a second identical call to `findFreeBlock`, which was not incorrect in this problem but potentially demonstrates unfamiliarity with the general pattern for using the result of a method call in both the Boolean condition of an `if` statement and within another clause of the `if` statement.

Across both parts, this Methods and Control question included many individual pieces, both in the conditional statements and in the iterations, where students were required to think carefully about edge cases and boundary conditions. It appears that many students particularly struggled with that aspect.

***What common student misconceptions or gaps in knowledge were seen in the responses to this question?***

| *Common Misconceptions/Knowledge Gaps*<br><br>*Write program code to create objects of a class and call methods.* | *Responses that Demonstrate Understanding* |
|---|---|
| Some responses added unnecessary type specifiers to the arguments in the method call.<br><br>`isMinuteFree(int period, int min)` | `isMinuteFree(period, min)` |
| Some responses incorrectly added class names to instance method calls.<br><br>`AppointmentBook.reserveBlock(p, m,`<br>`duration);` | `reserveBlock(p, m, duration);` |
| *Common Misconceptions/Knowledge Gaps*<br><br>*Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.* | *Responses that Demonstrate Understanding* |
| Some responses did not iterate over the exact specified range.<br><br>For part (a) considered too many minutes:<br>`for (int m = 0; m <= 60; m++)`<br><br>For part (b) did not consider all periods:<br>`for (int p = startPeriod; p < endPeriod;`<br>`p++)` | `for (int m = 0; m < 60; m++)`<br>`for (int m = 0; m <= 59; m++)`<br><br>`for (int p = startPeriod; p <= endPeriod;`<br>`p++)` |
| Some responses did not ensure that all of the elements in a range satisfied the specified condition, only the beginning and ending elements in the range.<br><br>`if (isMinuteFree(period, m) &&`<br>`isMinuteFree(period, m + duration))` | `for (int k = i; k < 60 && k < i + duration;`<br>`k++)`<br>`{`<br>`if (isMinuteFree(period, k))` |
| Some responses did not correctly use the return value from a method call to exclude the value `-1`.<br><br>`if (findFreeBlock(period, duration))` | `if (findFreeBlock(period, duration) != -1)` |

| | |
|---|---|
| or<br>`if (findFreeBlock(period, duration) > 0)` | or<br>`if (findFreeBlock(period, duration) >= 0)` |

***Based on your experience at the AP Reading with student responses, what advice would you offer teachers to help them improve student performance on the exam?***

- Practice not only the common counting-iteration pattern (start at 0, up to but excluding n) but also other common patterns (inclusive start/end bounds, start at n and continue a fixed number of times) as well as unusual ones.
- Reinforce the concept of edge cases and use concrete examples ("What if the parameters were 2 and 5, would the method work in this case?") to trace and test code to prevent off-by-one errors.
- Encourage the students, when stuck or going slowly on one problem, to set it aside for a moment and work on a different problem. Although this can be framed as test-taking strategy (which it also is), it is useful in general when stuck on programming problems (and other problems) outside of a test environment.

***What resources would you recommend to teachers to better prepare their students for the content and skill(s) required on this question?***

- Personal progress checks from units 2, 3, and 4 would be helpful to scaffold students' understanding for the Methods and Control free-response questions.
- The following AP Daily Videos and corresponding Topic Questions can be found in AP Classroom to support this Methods and Control free-response question:
  - *Write program code to create objects of a class and call methods.* Topics 2.3, 2.4, 2.5, and 2.7.
  - *Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.* Topics 3.2, 3.3, 3.4, 3.7, 4.2, and 4.3.

## Question 2

**Task:** Class Design
**Topic:** Sign
**Max Score:** 9
**Mean Score:** 4.74

***What were the responses to this question expected to demonstrate?***

This question tested the student's ability to:

- Write program code to define a new type by creating a class.
- Write program code to create objects of a class and call methods.
- Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

Students were asked to write a class named `Sign` that would contain a constructor and two methods. In implementing a solution, students were expected to demonstrate an understanding of class header, class constructor, and method header syntax. Students were expected to properly declare and initialize private instance variables to maintain the necessary information, typically the message to be delimited and the width of the lines in the sign.

The specification of the class required that two methods be implemented: `numberOfLines` needed to return the number of lines that the sign would have based on the message and the given width and `getLines` needed to return the delimited string, inserting a `";"` to break the given message into segments of a given width without inserting the `";"` on the final string segment. The details of these methods included somewhat more algorithmic complexity than some other Class Design problems have; in this case each of the specified methods required some algorithmic work, with no simple accessor or mutator methods needed for the design.

This question also tested the student's ability to work with string values. The student needed to understand how to use the `String` methods `substring` and `length` with correct method call syntax and arguments. The question also required the student to know how to check whether a `String` variable contained the empty string and to return `null` in that case.

***How well did the responses address the course content related to this question? How well did the responses integrate the skills required on this question?***

*Write program code to define a new type by creating a class.*

In a successful response, the headers for the class itself, the constructor, and both required methods were written correctly and declared as public, and private instance variables were declared and initialized in the constructor. Although it was possible to write a correct solution that pre-computes the results and stores them in instance variables, the majority of responses successfully took the approach of storing the values of the two constructor parameters for later use.

*Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements*

A successful solution for the `numberOfLines` method created a method that divided the message length to determine the number of lines in the sign based on the given sign width. This was typically implemented with one of two different algorithmic approaches. One approach used the length of the string and divided by the sign width using integer division. If the message length was not exactly divisible by the width, then the solution needed to add 1 for the partial line. The second approach used a loop to count how many sign widths the message length could contain. A correct solution again had to account for that partial last line and add 1. The solution also needed to account for the message length being less than the sign width, and for the message being an empty string.

A successful solution for the `getLines` method created a method that broke the string into smaller chunks according to the specified sign width and delimited the chunks with a `";"` to separate the lines (without putting an extra `";"` on the end of the constructed string). This was accomplished using a number of different valid approaches, both in terms of breaking up the larger string and in terms of deciding when to insert or concatenate the semicolons. Many solutions for this method were not correct due to incorrectly accounting for the last line, either adding an extra delimiter or attempting to access the string past its length (or both). This method also required checking for an empty string and returning `null` in that case. Most responses did not have this check. When the response included the check, it had to have the correct logic, which either checked if the length of the message was 0 or checked if the message was equal to the empty string.

*Write program code to create objects of a class and call methods.*

The method calls that students had to write in this problem were mostly limited to calling `String` methods. The only required method calls were to `length` and `substring`. Incorrect syntax was not a major source of student error, though, as mentioned above, many responses used the two-argument form of the `substring` method incorrectly, with an upper bound greater than the length of the string.

***What common student misconceptions or gaps in knowledge were seen in the responses to this question?***

| *Common Misconceptions/Knowledge Gaps*<br><br>*Write program code to define a new type by creating a class.* | *Responses that Demonstrate Understanding* |
|---|---|
| Some responses used incorrect syntax to declare the class (in some cases combining the class and constructor headers).<br><br>`public class Sign()`<br>or<br>`public class Sign(String m, int w)` | `public class Sign` |

| | |
|---|---|
| Some responses did not declare variables with `private` access<br><br>`public String message;`<br>`public int width;`<br>or<br>`String message;`<br>`int width;` | `private String message;`<br>`private int width;` |
| *Common Misconceptions/Knowledge Gaps*<br><br>*Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.* | *Responses that Demonstrate Understanding* |
| Some responses did not correctly account for the last line when counting lines.<br><br>`if (msg.length() > width)`<br>`{`<br>`   lines = msg / width;`<br>`}`<br>`else`<br>`{`<br>`   lines = 1;`<br>`}` | `if (msg.length() < width)`<br>`{`<br>`   lines = 1;`<br>`}`<br>`else if (msg.length() % width == 0)`<br>`{`<br>`   lines = msg.length() / width;`<br>`}`<br>`else`<br>`{`<br>`   lines = msg.length() / width + 1;`<br>`}`<br>or<br>`lines = msg.length() / width;`<br>`if (msg.length() % width != 0)`<br>`{`<br>`   lines++;`<br>`}` |
| Some responses failed to return `null` correctly when the message was empty:<br><br>`if (message.length() == 0)`<br>`{`<br>`   return "null";`<br>`}`<br><br>Some responses did not check for the empty string at all. | `if (message.length() == 0)`<br>`{`<br>`   return null;`<br>`}`<br>or<br>`if (message.equals(""))`<br>`{`<br>`   return null;`<br>`}` |

| | |
|---|---|
| Some responses did not handle the last line as a special case, typically including an extra semicolon and going out of bounds in the `substring` call.<br><br>`String result = "";`<br>`int x = 0;`<br><br>`while (x < msg.length())`<br>`{`<br>`   result += msg.substring(x, x + width) +`<br>`             ";";`<br>`   x += width;`<br>`}`<br>or<br>`String result = "";`<br>`int numLines = numberOfLines();`<br><br>`for (int i = 0; i < numLines; i++)`<br>`{`<br>`   result += msg.substring((i * width),`<br>`                 (i + 1) * width)) + ";";`<br>`}` | `String result = "";`<br>`int x = 0;`<br>`while (x + width < msg.length())`<br>`{`<br>`   result += msg.substring(x, x + width) +`<br>`             ";";`<br>`   x += width;`<br>`}`<br>`result += msg.substring(x);`<br>or<br>`String result = "";`<br>`int numLines = numberOfLines();`<br>`int i;`<br>`for (i = 0; i < numLines - 1; i++)`<br>`{`<br>`   result += msg.substring((i * width),`<br>`                 (i + 1) * width)) + ";";`<br>`}`<br>`result += msg.substring(i * width);`<br>or<br>`String result = "";`<br>`for (int i = 0; i < msg.length(); i++)`<br>`{`<br>`   result += msg.substring(i, i + 1);`<br>`   if ((i + 1) % width == 0 &&`<br>`       i != msg.length() - 1)`<br>`   {`<br>`      result += ";";`<br>`   }`<br>`}` |
| *Common Misconceptions/Knowledge Gaps*<br><br>*Write program code to create objects of a class and call methods.* | *Responses that Demonstrate Understanding* |
| Some responses did not call the `length` method to determine the number to divide.<br><br>`return msg / width;` | `return msg.length() / width;` |

***Based on your experience at the AP Reading with student responses, what advice would you offer teachers to help them improve student performance on the exam?***

- Reinforce the importance of looking for special cases in problem specification (such as the instruction to return `null` when the message is empty) and encourage handling the special case first.
- Practice with problems with edge cases, where the first or last element in a list or sequence will need special handling, either with a conditional statement inside the loop or by handling it outside the loop and adjusting the loop bounds. In particular, practice identifying the specific ways in which the edge case needs to be handled differently, which might not be explicitly explained in the problem statement.

***What resources would you recommend to teachers to better prepare their students for the content and skill(s) required on this question?***

- Personal progress checks from units 2 (String methods), 4 (String algorithms), and 5 would be helpful to scaffold students' understanding for the Class Design free-response questions that include inheritance.
- The following AP Daily Videos and corresponding Topic Questions can be found in AP Classroom to support this Class Design free-response question:
    - *Write program code to define a new type by creating a class.* Topics in unit 5.
    - *Write program code to create objects of a class and call methods.* Topics 2.6, 2.6, and 4.3.
    - *Write program code to satisfy methods using expressions, conditional statements, and iterative statements.* Topics 1.3, 3.2, 3.3, 3.4, and 3.5.

## Question 3

**Task:** Array / ArrayList
**Topic:** Weather Data
**Max Score:** 9
**Mean Score:** 5.07

***What were the responses to this question expected to demonstrate?***

This question tested the student's ability to:

- Write program code to create objects of a class and call methods.
- Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.
- Write program code to create, traverse, and manipulate elements in 1D array or ArrayList objects.

This question involved the traversal, analysis, and manipulation of an ArrayList object containing numbers modeling recorded temperatures. Students were expected to write two methods of the WeatherData class, using its ArrayList instance variable.

In part (a) students were expected to write a loop that accessed each element of the ArrayList instance variable temperatures and removed any elements that were outside of a range specified by the parameters upper and lower. Inside the loop, students were expected to use a conditional to identify elements for removal from the list and to call the remove method on the instance variable temperatures.

In part (b) students were asked to (1) identify the lengths of any heat waves of consecutive elements in temperatures that were greater than the parameter threshold, and (2) determine and return the length of the longest of the identified heat waves.

***How well did the responses address the course content related to this question? How well did the responses integrate the skills required on this question?***

*Write program code to create, traverse, and manipulate elements in 1D array or ArrayList objects.*

In part (a) responses generally iterated over the elements of the ArrayList using an indexed for loop. Some responses traversed the list in the forward direction and decremented the index variable to account for any elements that would be skipped after a removal. Other responses traversed the ArrayList in reverse order, which does not need the index variable adjustment. A common source of student error in this part was neglecting to compensate for the removed element while traversing. In part (b) responses usually iterated over the ArrayList using a for loop. Some responses used an indexed for loop and others used an enhanced for loop.

*Write program code to create objects of a class and call methods.*

In both parts (a) and (b) responses needed to use the `get` method to access values from `temperatures`. In part (a) responses also needed to use the `remove` method to remove elements from the list.

*Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.*

In part (a) responses typically used a compound conditional statement inside the loop to identify elements for removal from the list. Other responses used a multi-way selection (`if-else-if`) to identify elements for removal. In part (b) responses typically used an `if` statement that incremented an accumulator variable when an element of `temperatures` was above `threshold` and reset the accumulator when the end of a heat wave was identified. The responses also included a second variable to store the length of the longest heat wave found and a conditional to determine when the length of a heat wave exceeded the previous maximum length.

***What common student misconceptions or gaps in knowledge were seen in the responses to this question?***

| *Common Misconceptions/Knowledge Gaps*<br><br>*Write program code to create objects of a class and call methods.* | *Responses that Demonstrate Understanding* |
|---|---|
| Some responses did not call the `get` method to access elements of the list.<br><br>`if (temperatures[i] < lower \|\|`<br>`    temperatures[i] > upper)` | `if (temperatures.get(i) < lower \|\|`<br>`    temperatures.get(i) > upper)` |
| Some responses called the `remove` method incorrectly, such as calling it on the class name rather than the `ArrayList`.<br><br>`CleanData.remove(i);` | `temperatures.remove(i);` |
| *Common Misconceptions/Knowledge Gaps*<br><br>*Write program code to create, traverse, and manipulate elements in 1D or `ArrayList` objects.* | *Responses that Demonstrate Understanding* |

Some responses failed to access all elements of the list or accessed indices that were out-of-bounds.

```
int size = temperatures.size();
for (int i = size - 1; i > 0; i--)
{
   double t = temperatures.get(i);
   ...
}
```

or

```
int size = temperatures.size();
for (int i = size; i >= 0; i--)
{
   double t = temperatures.get(i);
   ...
}
```

```
int size = temperatures.size();
for (int i = size - 1; i >= 0; i--)
{
   double t = temperatures.get(i);
   ...
}
```

Some responses attempted to remove elements using an enhanced `for` loop.

```
for (double t : temperatures)
{
   if (t < lower || t > upper)
   {
      temperatures.remove(t);
   }
}
```

```
int i = 0;
while (i < temperatures.size())
{
   double t = temperatures.get(i);
   if (t < lower || t > upper)
   {
      temperatures.remove(i);
   }
   else
   {
      i++;
   }
}
```

*Common Misconceptions/Knowledge Gaps*

*Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.*

*Responses that Demonstrate Understanding*

Some responses failed to account for the shift in an `ArrayList` when an element is removed.

```
int size = temperatures.size();
for (int i = 0; i < size; i++)
{
   double t = temperatures.get(i);
   if (t < lower || t > upper)
   {
      temperatures.remove(i);
   }
}
```

```
int size = temperatures.size();
int i = 0;
while (i < size)
{
   double t = temperatures.get(i);
   if (t < lower || t > upper)
   {
      temperatures.remove(i);
   }
   else
   {
      i++;
   }
}
```

Some responses did not reset the variable used to find the length of each heat wave, effectively computing the number of days above `threshold` rather than the length of a heat wave.

```
int count = 0;
int max = 0;
for (double temp : temperatures)
{
   if (temp > threshold)
   {
      count++;
   }
   if (count > max)
   {
      max = count;
   }
}
```

```
int count = 0;
int max = 0;
for (double temp : temperatures)
{
   if (temp > threshold)
   {
      count++;
   }
   else
   {
      count = 0;
   }
   if (count > max)
   {
      max = count;
   }
}
```

Some responses did not account for the situation where the maximum heat wave occurs at the end of the list.

```
int count = 0;
int max = 0;
for (double temp : temperatures)
{
   if (temp > threshold)
   {
      count ++;
   }
   else
   {
      if (count > max)
      {
         max = count;
      }
      count = 0;
   }
}
```

```
int count = 0;
int max = 0;
for (double temp : temperatures)
{
   if (temp > threshold)
   {
      count ++;
   }
   else
   {
      if (count > max)
      {
         max = count;
      }
      count = 0;
   }
}
if (count > max)
{
   max = count;
}
```

***Based on your experience at the AP Reading with student responses, what advice would you offer teachers to help them improve student performance on the exam?***

- Reinforce the difference between `ArrayList` objects and arrays, specifically when it comes to accessing elements from each.
- Reinforce the consideration of edge cases that might make code fail to execute as intended.
- Emphasize the scenarios for which an enhanced `for` loop should not be used.
- Continue practicing creating loop headers to traverse a collection.
  - Practice with loops traversing in a forward direction
  - Practice with loops traversing in a reverse direction

    - Practice with enhanced `for` loops
- Continue practicing logic to determine a maximum value

***What resources would you recommend to teachers to better prepare their students for the content and skill(s) required on this question?***

- Personal progress checks from unit 7 would be helpful to scaffold students' understanding for the `ArrayList` free-response questions.
- The following AP Daily Videos and corresponding Topic Questions can be found in AP Classroom to support this `ArrayList` free-response question:
    - *Write program code to create objects of a class and call methods.* Topics 2.3, 2.4, and 2.5.
    - *Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.* Topics 3.2, 3.3, 3.4, and 3.5.
    - *Write program code to create, traverse, and manipulate elements in 1D or ArrayList objects.* Topics 7.2, 7.3, 7.4, and 7.5.

## Question 4

**Task:** 2D Array
**Topic:** Candy Box
**Max Score:** 9
**Mean Score:** 3.43

***What were the responses to this question expected to demonstrate?***

This question tested the student's ability to:

- Write program code to create objects of a class and call methods.
- Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.
- Write program code to create, traverse, and manipulate elements in 2D array objects.

This question involved the manipulation of a two-dimensional (2D) array of `Candy` references and interactions between two classes. The `Candy` class included one method, `getFlavor`. A second class, `BoxOfCandy`, included an instance variable, `box`, a 2D array of `Candy` references. In addition, the class contained two methods, one to be written in part (a) and one to be written in part (b).

In part (a) students were asked to write a non-void method, `moveCandyToFirstRow`, which had one integer parameter, `col`, and needed to implement two interacting algorithms. The first algorithm moves a piece of candy, if necessary and possible, to the first row of `col`. Moving a piece of candy is deemed *necessary* if the reference to the element of `box` at row `0` column `col` is `null`, and it is *possible* if there is a `Candy` object (non-`null`) in column `col`. As a result of the second algorithm, students were expected to return `true` if there is a piece of candy in the first row of `col` upon exit—either because it was already there or because it was moved there—and `false` otherwise.

In part (b) students were asked to write a method called `removeNextByFlavor` that returns the first `Candy` object that matches the `flavor` parameter, searching the 2D array in a specified order, or returns `null` if there is not any piece of candy with the given flavor. If a piece of candy matching the flavor is found, the student was also expected to remove the reference to that object from the 2D array.

***How well did the responses address the course content related to this question? How well did the responses integrate the skills required on this question?***

*Write program code to create objects of a class and call methods*

In part (a) there were not any method calls required. In part (b) most responses were able to call the `getFlavor` method successfully. However, some responses tried to make new instances of `Candy` objects. Since no information was given about what constructors are implemented in the `Candy` class, this code was interacting with parts of the `Candy` class that are not guaranteed to exist.

*Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements*

While most responses could write code with conditional statements, a large majority of responses did not correctly compare objects with `null` using the `==` or `!=` operators: many responses failed to check whether a `Candy` reference was `null` before calling its `getFlavor` method. Many responses included code meant to compare `String` objects as required, but some incorrectly used `==` rather than the `equals` method to make the comparison.

*Write program code to create, traverse, and manipulate elements in 2D array objects.*

Many responses successfully traversed a column in part (a), but some responses confused the check for number of rows and used the number of columns as the loop bound. In part (b) the students were expected to do a specific traversal of a 2D array, starting with the last row, traversing from left to right, and then moving up a row until the search was complete. Many responses incorrectly traversed the array with a standard traversal from the top down. Often the responses that attempted the specified traversal had bounds errors.

***What common student misconceptions or gaps in knowledge were seen in the responses to this question?***

| *Common Misconceptions/Knowledge Gaps*<br><br>*Write program code to create objects of a class and call methods.* | *Responses that Demonstrate Understanding* |
|---|---|
| Some responses attempted to use constructors from the `Candy` class that were not given.<br><br>`Candy c = new Candy();`<br>`c = box[r][c];` | `Candy c;`<br>`c = box[r][c];`<br>or<br>`Candy c = null;`<br>`c = box[r][c];`<br>or<br>`Candy c = box[r][c];` |
| Some responses did not correctly compare `String` values.<br><br>`if (box[r][c].getFlavor() == flavor)` | `if (box[r][c].getFlavor().equals(flavor))` |
| *Common Misconceptions/Knowledge Gaps*<br><br>*Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.* | *Responses that Demonstrate Understanding* |

| | |
|---|---|
| Some responses did not correctly compare a `Candy` object with `null`.<br><br>`if (!(box[0][col].equals(null)))`<br>or<br>`if (box[0][col].equals(null))`<br>or<br>`if (box[0][col] != "null")`<br>or<br>`if (box[0][col] == "null")` | `if (box[0][col] != null)`<br>or<br>`if (box[0][col] == null)` |
| Some responses did not correctly check for `null` before trying to access the flavor of the candy.<br><br>`if(box[r][c].getFlavor().equals(flavor))` | `if (box[r][c] != null &&`<br>`    box[r][c].getFlavor().equals(flavor))`<br>or<br>`if (box[r][c] != null)`<br>`{`<br>`   if (box[r][c].getFlavor()`<br>`                .equals(flavor))` |
| *Common Misconceptions/Knowledge Gaps*<br><br>*Write program code to create, traverse, and manipulate elements in 2D array objects.* | *Responses that Demonstrate Understanding* |
| Some responses did not correctly traverse a column of a 2D array.<br><br>`for (int k = 0; k < box[0].length; k++)`<br>`{`<br>`   if (box[k][col] != null)` | `for (int k = 0; k < box.length; k++)`<br>`{`<br>`   if (box[k][col] != null)` |
| Some responses did not correctly traverse the 2D array starting with the last row traversing from left to right, then the next-to-last row traversing from left to right, etc.<br><br>`for (int r = 0; r < box.length; r++)`<br>`{`<br>`   for (int c = 0; c < box[0].length; c++)`<br>or<br>`for (int r = box.length; r > 0; r--)`<br>`{`<br>`   for (int c = 0; c < box[0].length; c++)`<br>or<br>`for (int r = box.length - 1; r >= 0; r++)`<br>`{`<br>`   for (int c = 0; c < box[0].length; c++)`<br>or<br>`for (int r = box.length - 1; r >= 0; r--)`<br>`{`<br>`   for (int c = box[0].length - 1; c >= 0;`<br>`        c--)` | `for (int r = box.length - 1; r >= 0; r--)`<br>`{`<br>`   for (int c = 0; c < box[0].length; c++)` |

***Based on your experience at the AP Reading with student responses, what advice would you offer teachers to help them improve student performance on the exam?***

- Practice with some problems where some of the data in a list or array may be `null` values, to reinforce the importance of guarding method calls on those values to prevent exceptions.
- Consider using physical models to represent objects. An easy example of how to do this is to use different size sticky notes. The large ones are the objects. Then use smaller sticky notes on each large note to represent the attributes and methods. Tape the private variables on the large note so that they cannot be accessed by others. Then show how you can call on the public methods to access information.
- Consider using physical models of 2D arrays to model different types of traversals. Some examples of models include multi-week pill boxes, board games that use a grid, or calculator/phone holders in the classroom. Model alternate types of traversals. After students understand the different variations, have them write the code and test their programs. Let the students come up with their own interesting variations for traversing a 2D array.
- Encourage the students, when sitting down to work on multiple problems or projects, to look through all of them first to make decisions about where to start and in what order to approach them. Although this can be framed as test-taking strategy (which it also is), it is useful in general when planning how to concurrently work on multiple programming problems (and other problems) outside of a test environment.

***What resources would you recommend to teachers to better prepare their students for the content and skill(s) required on this question?***

- Personal progress checks from Unit 8 would be helpful to scaffold students' understanding for the 2D array free response questions.
- The following AP Daily Videos and corresponding Topic Questions can be found in AP Classroom to support this 2D array free response question.
  - *Write program code to create objects of a class and call methods.* Topics 2.2 and 2.3.
  - *Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.* Topics 3.1, 3.3, 3.5, and 3.7.
  - *Write program code to create, traverse, and manipulate elements in 2D array objects.* Topics 8.1 and 8.2.