Chief Reader Report on Student Responses:

2019 AP® Computer Science A Free-Response Questions

- Number of Students Scored 69,685
- Number of Readers 324
- Score Distribution

Exam Score	N	%At
5	18,583	26.7
4	15,275	21.9
3	14,660	21.0
2	8,320	11.9
1	12,847	18.4

- Global Mean 3.26

The following comments on the 2019 free-response questions for AP® Computer Science A were written by the Chief Reader, John Cigas, Professor, Park 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:** Calendar

Max. Points: 9 **Mean Score:** 6.18

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; and
- Write program code to satisfy methods using expressions, conditional statements, and iterative statements.

More specifically, this question assessed the ability to use numeric primitive types, iterate through a range, call static methods, and use a method's return value in a conditional expression.

In part (a) students were asked to count the number of years within a given range (inclusive) that were leap years. They were provided the method `isLeapYear` to determine whether a particular year was a leap year and were instructed to call this method rather than implementing the (unspecified) leap year criteria. They were expected to initialize a numeric counter, iterate through all years in the range, call the given method on each year, use the result to conditionally update the counter, and return the counter after the iteration.

In part (b) students were asked to determine the day of the week on which a given date (month, day, and year) falls. They were provided the method `firstDayOfYear` to determine the day of the week of the first day of a year, encoded 0 through 6. They were also provided the method `dayOfYear` to determine the ordinal date (the number of days of the year that have elapsed, including the given day), from 1 through 366. The students were instructed to call these methods rather than implementing the (unspecified) logic to compute them.

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 methods. In part (a) responses called a static method `isLeapYear` within the context of a loop. In part (b) responses called two static methods, `firstDayOfYear` and `dayOfYear`, the latter with multiple parameters, that each returned an integer. In addition, the parameters passed to these methods were themselves the parameters passed into the method. Most responses successfully called the methods with the proper parameters and then used the returned results appropriately.

Write program code to satisfy methods using expressions, conditional statements, and iterative statements.

In part (a) students wrote a loop with the lower and upper bounds specified by parameters to the method. Within the loop, they wrote a conditional expression to update a counter. After the loop, they returned the calculated value. The majority of students did this correctly, although some failed to include the upper bound as part of the calculation. The most common issue was making assumptions about what is and is not a leap year and writing code based on those assumptions, instead of calling `isLeapYear` within the loop.

In part (b) students used the result of two method calls to perform a calculation, with the final result constrained to be in the range 0 through 6, and then return that calculated value. This calculation proved to be challenging as many students were not able to interpret the results of the method calls correctly or were unable to constrain the final value to be in the specified range.

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

Common Misconceptions/Knowledge Gaps *Write program code to create objects of a class and call methods.*	*Responses that Demonstrate Understanding*

Students made syntactically incorrect calls to `isLeapYear`, including misplacement or omission of the argument.
 	
`if (y.isLeapYear())…`
`if (isLeapYear())…`
`if (isLeapYear)…`
 	
Students treated the value returned from `isLeapYear` as something other than a `boolean`.
 	
`if (isLeapYear(y).equals("true"))…`
`if (isLeapYear(y).equals(true))…`
 	
Students attempted their own check for leap year rather than using the method provided.
 	
`if (y % 4 == 0)…`
	`if (isLeapYear(y))…`
	`if (isLeapYear(y) == true)…`

Students added unnecessary type specifiers to the arguments in the function call.
 	
`int dayNumber =`
`   dayOfYear(int month, int day,`
`             int year);`
	`int dayNumber = dayOfYear(month, day, year);`

Students confused static method calls with constructors or instance method calls.
 	
`int firstDay = new firstDayOfYear(year);`
`int firstDay =`
`   this.firstDayOfYear(year);`
	`int firstDay = firstDayOfYear(year);`

Common Misconceptions/Knowledge Gaps *Write program code to satisfy methods using expressions, conditional statements, and iterative statements.*	*Responses that Demonstrate Understanding*
Students did not initialize local variables before use `int count;` or did not declare a local variable at all.	`int count = 0;`

Students did not consider all years in the specified range.
 	
`for (int y = year1; y < year2; y++)`
 	
`for (int y = year1 + 1; y <= year2; y++)`
	`for (int y = year1; y <= year2; y++)`

Students looped over the length of the range, rather than the years in the range, but did not adjust for these loop bounds when calling `isLeapYear`.

```
int diff = year2 - year1;
for (int y = 0; y <= diff; y++)
{
   if (isLeapYear(y))
   {
      count++;
   }
}
```

```
int diff = year2 - year1;
for (int y = 0; y <= diff; y++)
{
   if (isLeapYear(year1 + y))
   {
      count++;
   }
}
```

Students returned early inside the loop instead of returning after checking all the years in the range.

```
for (int y = year1; y <= year2; y++)
{
   if (isLeapYear(y))
   {
      count++;
   }
   else
   {
      return count;
   }
}
```

```
for (int y = year1; y <= year2; y++)
{
   if (isLeapYear(y))
   {
      count++;
   }
}
return count;
```

Students did not account for day counts being numbered 1-366 and days being numbered 0-6.

```
return (firstDayOfYear(year) +
   dayOfYear(month, day, year)) % 7;
```

Students did not ensure that the final value is within range 0-6, due to modulo on a partial result rather than the final result, or not accounting for order of operations.

```
return firstDayOfYear(year) +
   dayOfYear(month, day, year) % 7 - 1;

return firstDayOfYear(year) - 1 +
   dayOfYear(month, day, year) % 7;
```

```
return (firstDayOfYear(year) +
   dayOfYear(month, day, year) - 1) % 7;
```

Students looped to compute the day of the week, but used incorrect loop sustaining conditions.

```
d = firstDayOfYear(year) +
    dayOfYear(month, day, year) - 1;
while (d > 7)
{
   d -= 7;
}
```

```
d = firstDayOfYear(year) +
    dayOfYear(month, day, year) - 1;
while (d >= 7)
{
   d -= 7;
}
```

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

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

- Students need to practice invoking and writing static methods.
    - Assign problems requiring students to invoke static methods.
    - Assign programs requiring students to write static methods.
- Students need to practice using provided methods to solve problems.
    - Assign problems requiring students to invoke methods that they didn't implement.

Write program code to satisfy methods using expressions, conditional statements, and iterative statements.

- Students need to practice iterating over a range of integers unrelated to a collection.
    - Assign problems that require a loop to start at some integer besides 0.
    - Assign problems that require inclusion/exclusion of the specified boundary conditions.
- Students need to practice calculations involving the `%` operator.
    - Assign problems that require constraining values to a certain range, like minutes in an hour or hours in a day.
- Students need to practice writing conditionals that do not require an `else` part.
    - Assign problems that require a return when an item is found in sequential search.
    - Assign problems that require a return when an item is not present or not found during a search.
    - Assign problems that require initializing an empty `String` to some other default value.

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

Suggested resources include:

The skills that students find most difficult to master are only learned over time and require repeated practice. The Personal Progress Checks at the end of each unit allow teachers and students to gauge student understanding of the content and skills taught in that unit. In particular, the Methods and Control Structures free-response questions that are included in the Personal Progress Checks in Units 2, 3, 4, and 10 provide multiple opportunities for students to practice writing program code to create objects of a class and call methods, along with writing program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

The Unit 2 Personal Progress Check, as well as the formative Topic Questions in Topics 2.3-2.5 meant to be assigned as the unit is taught, will provide the most practice with invoking methods. Conditional statements are addressed specifically in Unit 3 but will also be found in Unit 4. Teachers should make sure that students get repeated exposure to these types of questions. For practice with loops and boundary conditions, the Personal Progress Check in Unit 4 and the Topic Questions for Topics 4.1 and 4.2 will provide the most practice for students.

For additional practice with these skills on summative items that are similar to this type of question, the AP Question Bank can be filtered to find questions that address skills 3.A and 3.C and focus on content from Units 2-4. However, because the Question Bank contains items from previous AP Exams and Official Practice Exams, these items should be previewed before assigning them to students to ensure that students are ready to apply all skills and content knowledge that the items require.

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

- The 2018 free-response question 1, Frog Simulation, requires students to write a method using calls to provided methods, as well as loops and conditionals for control: https://secure-media.collegeboard.org/ap/pdf/ap18-frq-computer-science-a.pdf

- The 2017 free-response question 3, PhraseEditor, requires students to call methods of the newly presented class. This provides students with practice calling methods that were not studied in class. This resource can be found here: https://apcentral.collegeboard.org/pdf/ap-computer-science-a-frq-2017.pdf
- The Practice-It! website hosted by the University of Washington offers practice for students to analyze program code that calls methods. This practice is consolidated into Chapter 3. This resource can be found here: https://practiceit.cs.washington.edu/problem/list

Write program code to satisfy methods using expressions, conditional statements, and iterative statements.

- The Runestone Interactive Java Review offers an interactive environment for students to practice course content. Specifically, the Java Basics: Classes and Objects section and the Object-Oriented Concepts section would be most helpful for this question. This resource can be found here: http://interactivepython.org/runestone/static/JavaReview/index.html

Question #2 **Task:** Class Design **Topic:** Step Tracker

Max. Points: 9 **Mean Score:** 4.90

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; and
- Write program code to satisfy methods using expressions, conditional statements, and iterative statements.

Students were asked to design the class `StepTracker`, which implements a fitness tracking system. Students were expected to demonstrate an understanding of class constructor and method header syntax. Additionally, students were expected to determine the data types and number of instance variables needed to track the information shown in the example. Students were then expected to correctly declare, initialize, access, and update the instance variables. Students were expected to properly protect their data members by declaring them as `private` and to properly define the methods `addDailySteps`, `activeDays`, and `averageSteps`. Students also had to recognize the need for floating-point division when using integers to calculate the average in `averageSteps`.

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.

Successful response designs typically followed one of the following two strategies:

- Declare four numeric instance variables to hold the active steps threshold, as well as track the total number of steps, days, and active days, then update these as appropriate in the method `addDailySteps`.
- Declare an `ArrayList` to hold each day's step count and a numeric instance variable to hold the active steps threshold. The method `addDailySteps` appended a count to the `ArrayList`. The other methods iterated through the `ArrayList` to calculate a value each time the methods were called.

While many responses used one of the overall design strategies, they often made mistakes with the structural implementation details.

Write program code to satisfy methods using expressions, conditional statements, and iterative statements.

When responses had an appropriate design strategy, they generally had reasonable ways of implementing the methods. The most frequent problems were not calculating an average as a `double`, not checking for active days appropriately, and not updating all instance variables in `addDailySteps`.

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

Common Misconceptions/Knowledge Gaps *Write program code to define a new type by creating a class.*	*Responses that Demonstrate Understanding*
Students omitted the keyword `private` when declaring an instance variable. `int numActiveDays;`	`private int numActiveDays;`

Students specified the wrong number of parameters to the class constructor.
 	
`public StepTracker()`
`public StepTracker(int threshold,`
`                   int numDays)`
	`public StepTracker(int threshold)`

Students added a return type to the `addDailySteps` method.
 	
`public void int addDailySteps(int steps)`
`public int addDailySteps(int steps)`
	`public void addDailySteps(int steps)`

Students omitted the return type for the methods `activeDays` or `averageSteps`.
 	
`public void activeDays()`
`public averageSteps()`
 	
Students added a parameter to the `activeDays` or `averageSteps` methods.
 	
`public int activeDays(int numSteps)`
`public double averageSteps(int numSteps)`
 	
Students used the `private` modifier in a method header.
 	
`private int activeDays()`
`private double averageSteps()`
	`public int activeDays()`
	`public double averageSteps()`

Students used `int` for the return type of `averageSteps` instead of `double`. `public int averageSteps()`	`public double averageSteps()`

Students compared the daily step count to some numeric constant instead of an instance variable.
 	
`public void addDailySteps(int steps)`
`{`
`   if (steps >= 10000)`
`   {`
`      numActiveDays++;`
`   }`
`   totalSteps += steps;`
`   numDays++;`
`}`
 	
Students omitted updating some instance variables.
 	
`public void addDailySteps(int steps)`
`{`
`   if (steps >= minSteps)`
`   {`
`      numActiveDays++;`
`   }`
`}`
	`public void addDailySteps(int steps)`
	`{`
	`   if (steps >= minSteps)`
	`   {`
	`      numActiveDays++;`
	`   }`
	`   totalSteps += steps;`
	`   numDays++;`
	`}`

Students placed code outside of the class.
 	
`private int numActiveDays;`
 	
`public class StepTracker()`
`{`
`   …`
`}`
	`public class StepTracker()`
	`{`
	`   private int numActiveDays;`
	`   …`
	`}`

Common Misconceptions/Knowledge Gaps *Write program code to satisfy methods using expressions, conditional statements, and iterative statements.*	*Responses that Demonstrate Understanding*

Students omitted the lower bound when checking for an active day.
 	
`public void addDailySteps(int steps)`
`{`
`   if (steps > minSteps)`
`   {`
`      numActiveDays++;`
`   }`
`   totalSteps += steps;`
`   numDays++;`
`}`
	`public void addDailySteps(int steps)`
	`{`
	`   if (steps >= minSteps)`
	`   {`
	`      numActiveDays++;`
	`   }`
	`   totalSteps += steps;`
	`   numDays++;`
	`}`

Students performed integer division.
 	
`public double averageSteps()`
`{`
`   if (numDays == 0)`
`   {`
`      return 0.0;`
`   }`
`   else`
`   {`
`      return totalSteps / numDays;`
`   }`
`}`
 	
Students performed integer division before casting.
 	
`public double averageSteps()`
`{`
`   if (numDays == 0)`
`   {`
`      return 0.0;`
`   }`
`   else`
`   {`
`      return (double)`
`         (totalSteps / numDays);`
`   }`
`}`
	`public double averageSteps()`
	`{`
	`   if (numDays == 0)`
	`   {`
	`      return 0.0;`
	`   }`
	`   else`
	`   {`
	`      return (double) totalSteps / numDays;`
	`   }`
	`}`

Students failed to check for potential division by zero.
 	
`public double averageSteps()`
`{`
`   return (double) totalSteps / numDays;`
`}`	

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

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

- Students need to practice determining the instance variables needed to maintain the state of an object in a class.
- Students need to practice designating `private` visibility for instance variables.
- Students need to practice defining instance variables for attributes to be initialized by a constructor.
    - Assign problems that require a parameter when creating a new object of a class.
- Students need to practice defining behaviors of an object through void methods, with and without parameters.
- Students need to practice defining behaviors of an object through non-void methods, with and without parameters.
- Students need to practice designating `public` visibility for methods invoked by objects outside a class.

Write program code to satisfy methods using expressions, conditional statements, and iterative statements.

- Students need to recognize when integer division is not appropriate.
    - Assign problems like calculating an average or a ratio.
- Students need to practice using parameters rather than hardcoding constants.
- Students need to recognize when there can be an invalid calculation.
    - Assign problems involving division where the divisor could be 0.

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

Suggested resources include:

The Class free-response questions that are included in the Personal Progress Checks in Units 5 and 9 provide two opportunities for students to practice writing program code to define a new type by creating a class, along with writing program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

The Unit 5 Personal Progress Check, as well as the formative Topic Questions meant to be assigned as the unit is taught, will provide the most practice with defining a new type, including both attributes and behaviors. The Topic Questions in Topics 5.1 and 5.2 cover determining and defining instance variables, while the Topic Questions in Topics 5.4-5.8 deal with defining the behaviors of a class through different types of methods.

For additional practice with these skills on summative items that are similar to this type of question, the AP Question Bank can be filtered to find questions that address skills 3.B and 3.C and focus on content from Unit 5. Class free-response questions that are tagged to Unit 9 will be similar, however they will cover the additional content of inheritance. Because the Question Bank contains items from previous AP Exams and Official Practice Exams, these items should be previewed before assigning them to students to ensure that students are ready to apply all necessary skills and content knowledge that the items require.

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

- Data Lab and Celebrity Lab are two College Board provided labs that contain specific activities for students to practice defining a new type by creating a class.

- The Runestone Interactive Java Review offers an interactive environment for students to practice course content. Specifically, the Java Basics: Classes and Objects section and the Object-Oriented Concepts section would be most helpful for this question. This resource can be found here: http://interactivepython.org/runestone/static/JavaReview/index.html
- The Practice-It! website hosted by the University of Washington offers practice for students to analyze program code that calls methods. This practice is consolidated into Chapter 3. In addition, Chapter 8: Classes offers students practice in completing classes by writing methods. This resource can be found here: https://practiceit.cs.washington.edu/problem/list

Question #3 **Task:** `ArrayList` Processing **Topic:** Delimiters

Max. Points: 9 **Mean Score:** 5.56

What were the responses to this question expected to demonstrate?

This question tested the student's ability to:

- Write program code to satisfy methods using expressions, conditional statements, and iterative statements; and
- Write program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.

This question involved manipulation of both a one-dimensional array and an `ArrayList`, both containing `String` values. Students were expected to write two methods in the enclosing `Delimiters` class, making use of two instance variables of type `String`.

In part (a) students were expected to create an `ArrayList` of `String` objects, then add to it all the values from the given array that matched either of two instance variables. Students had to construct a new `ArrayList` and write a loop to access each element of an array parameter. Inside the loop, students were expected to compare each `String` value in the array with each of the two instance variables and add each matching `String` value to the constructed `ArrayList`.

In part (b) students were given an `ArrayList` containing `String` objects representing open and close delimiters. Students were asked to develop an algorithm to determine whether the given `ArrayList` represents a balanced sequence of open and close delimiters. A sequence is balanced when two conditions are met: (1) When traversing the `ArrayList` from the first element to the last element, there is no point at which there are more close delimiters than open delimiters. (2) The total number of open delimiters is equal to the total number of close delimiters. Students had to write a loop to access each element of the given `ArrayList`. Inside the loop, students had to compare each `String` value in the `ArrayList` to the instance variables and then update accumulator(s) appropriately.

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 satisfy methods using expressions, conditional statements, and iterative statements.

Some responses failed to compare `String` variables correctly, using the `==` operator, `contains` method, or `indexOf` method instead of the `equals` or `compareTo` methods.

Most responses used conditional statements appropriately to determine when two `String` variables matched and then took an appropriate action. Responses were less proficient at implementing conditional statements to identify when delimiters were balanced or unbalanced. Some responses neglected to initialize the variables used as counters.

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

While most responses could add an identified `String` to an `ArrayList`, many did not create a new `ArrayList` first, or created it incorrectly. Similarly, most responses iterated though the array and `ArrayList`, but some confused access to array elements with access to `ArrayList` elements.

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

Common Misconceptions/Knowledge Gaps *Write program code to satisfy methods using expressions, conditional statements, and iterative statements.*	*Responses that Demonstrate Understanding*

Students used the `==` operator to compare `String` values for equality.
 	
`if (str == openDel)…`
 	
Students used the `indexOf` or `contains` method to compare `String` values for equality
 	
`if (str.indexOf(openDel) != -1)…`
`if (str.contains(openDel))…`
	`if (str.equals(openDel))…`
	`if (str.compareTo(openDel) == 0)…`

Students compared elements to example strings instead of `openDel` and `closeDel` instance variables.
 	
`if (str.equals("<q>"))…`
	`if (str.equals(openDel))…`
	`if (str.compareTo(openDel) == 0)…`

Students failed to initialize accumulators to zero.
 	
`int count;`
 	
`for (String str : delimiters)`
`{`
`   if (str.equals(openDel))`
`   {`
`      count++;`
`   }`
`   else`
`   {`
`      count--;`
`      …`
`   }`
`}`
 	
Students initialized accumulators inside the loop.
 	
`for (String str : delimiters)`
`{`
`   int count = 0;`
 	
`   if (str.equals(openDel))`
`   {`
`      count++;`
`   }`
`   else`
`   {`
`      count--;`
`      …`
`   }`
`}`
	`int count = 0;`
	
	`for (String str : delimiters)`
	`{`
	`   if (str.equals(openDel))`
	`   {`
	`      count++;`
	`   }`
	`   else`
	`   {`
	`      count--;`
	`      …`
	`   }`
	`}`

Students returned `true` before accessing all necessary elements of `delimiters`.

```
int count = 0;

for (String str : delimiters)
{
   if (str.equals(openDel))
   {
      count++;
   }
   else
   {
      count--;
   }
   if (count < 0)
   {
      return false;
   }
   else
   {
      return true;
   }
}
return count == 0;
```

Students did not return a `boolean` value in all cases.

```
int count = 0;

for (String str : delimiters)
{
   if (str.equals(openDel))
   {
      count++;
   }
   else
   {
      count--;
   }
   if (count < 0)
   {
      return false;
   }
}

if (count == 0)
{
   return true;
}
```

```
int count = 0;

for (String str : delimiters)
{
   if (str.equals(openDel))
   {
      count++;
   }
   else
   {
      count--;
   }
   if (count < 0)
   {
      return false;
   }
}

return count == 0;
```

Students omited a test for more close delimiters than open delimiters inside the loop.

```
int openCount = 0;
int closeCount = 0;

for (String str : delimiters)
{
   if (str.equals(openDel))
```

```
int openCount = 0;
int closeCount = 0;

for (String str : delimiters)
{
   if (str.equals(openDel))
   {
      openCount++;
   }
```

`   {`
`      openCount++;`
`   }`
`   else`
`   {`
`      closeCount++;`
`   }`
`}`
 	
`return openCount == closeCount;`
	`   else`
	`   {`
	`      closeCount++;`
	`   }`
	`   if (closeCount > openCount)`
	`   {`
	`      return false;`
	`   }`
	`}`
	
	`return openCount == closeCount;`

Students tested if the number of open and close delimiters is the same inside the loop.
 	
`int openCount = 0;`
`int closeCount = 0;`
 	
`for (String str : delimiters)`
`{`
`   if (str.equals(openDel))`
`   {`
`      openCount++;`
`   }`
`   else`
`   {`
`      closeCount++;`
`   }`
 	
`   if (closeCount > openCount)`
`   {`
`      return false;`
`   }`
 	
`   if (openCount == closeCount)`
`   {`
`      return true;`
`   }`
`   else`
`   {`
`      return false;`
`   }`
`}`
	`int openCount = 0;`
	`int closeCount = 0;`
	
	`for (String str : delimiters)`
	`{`
	`   if (str.equals(openDel))`
	`   {`
	`      openCount++;`
	`   }`
	`   else`
	`   {`
	`      closeCount++;`
	`   }`
	
	`   if (closeCount > openCount)`
	`   {`
	`      return false;`
	`   }`
	`}`
	
	`if (openCount == closeCount)`
	`{`
	`   return true;`
	`}`
	`else`
	`{`
	`   return false;`
	`}`

Common Misconceptions/Knowledge Gaps *Write program code to create, traverse, and manipulate elements in 1D array or* `ArrayList` *objects.*	*Responses that Demonstrate Understanding*

Students declared an `ArrayList` without initializing it
 	
`ArrayList<String> d;`
 	
or did not declare an `ArrayList` at all.
 	
Students constructed an `ArrayList` using incorrect syntax.
 	
`d = ArrayList<String>();`
`int d = new ArrayList<String>();`
	`ArrayList<String> d;`
	`d = new ArrayList<String>();`

`new ArrayList<String>() d;`	

Students accessed elements of the `tokens` array as if from an `ArrayList`.
 	
`for (int i = 0; i < tokens.size(); i++)`
`{`
`   if (tokens.get(i).equals(openDel) ||`
`        tokens.get(i).equals(closeDel))`
`   {`
`      d.add(tokens.get(i));`
`   }`
`}`
 	
Students did not access all elements of the `tokens` array.
 	
`for (int i = 0; i < tokens.length - 1; i++)`
`{`
`   if (tokens[i].equals(openDel) ||`
`        tokens[i].equals(closeDel))`
`   {`
`      d.add(tokens[i]);`
`   }`
`}`
 	
Students accessed elements outside the bounds of the `tokens` array.
 	
`for (int i = 0; i <= tokens.length; i++)`
`{`
`   if (tokens[i].equals(openDel) ||`
`        tokens[i].equals(closeDel))`
`   {`
`      d.add(tokens[i]);`
`   }`
`}`
 	
Students added tokens to the `ArrayList` at potentially nonexistent indices.
 	
`for (int i = 0; i < tokens.length; i++)`
`{`
`   if (tokens[i].equals(openDel) ||`
`        tokens[i].equals(closeDel))`
`   {`
`      d.add(i, tokens[i]);`
`   }`
`}`
	`for (int i = 0; i < tokens.length; i++)`
	`{`
	`   if (tokens[i].equals(openDel) ||`
	`        tokens[i].equals(closeDel))`
	`   {`
	`      d.add(tokens[i]);`
	`   }`
	`}`

Students accessed elements of the `ArrayList` as if from an array.
 	
`for (int i = 0; i < delimiters.length; i++)`
`{`
`   if (delimiters[i].equals(openDel))`
	`for (int i = 0; i < delimeters.size();`
	`     i++)`
	`{`
	`   if (delimiters.get(i).equals(openDel))`

```
   {
      count++;
   }
   else
   {
      count--;
   }
   …
}
```

```
   {
      count++;
   }
   else
   {
      count--;
   }
   …
}
```

Students accessed the entire array or `ArrayList` instead of a specific element.

```
if (tokens.equals(openDel))…
```

```
if (tokens[i].equals(openDel))…
```

Students did not use the specified parameters when accessing the array or `ArrayList`.

```
for (String str : getDelimitersList)
```

```
for (String str : tokens)
```

Students used the names of parameters from the constructor instead of the names of instance variables.

```
for (String str : tokens)
{
   if (str.equals(open) ||
       str.equals(close))
   {
      d.add(str);
   }
}
```

```
for (String str : tokens)
{
   if (str.equals(openDel) ||
       str.equals(closeDel))
   {
      d.add(str);
   }
}
```

Students did not maintain the original delimiter order.

```
for (String str : tokens)
{
   if (str.equals(openDel) ||
       str.equals(closeDel))
   {
      d.add(0, str);
   }
}
```

```
for (String str : tokens)
{
   if (str.equals(openDel) ||
       str.equals(closeDel))
   {
      d.add(str);
   }
}
```

Students removed elements of the `ArrayList`, destroying persistent data.

```
while (delemeters.size() > 0)
{
   if (delimiters.get(0).equals(openDel))
   {
      count++;
   }
   else
   {
      count--;
   }

   delimiters.remove(0);

   if (count < 0)
```

```
for (int i = 0; i < delemeters.size();
     i++)
{
   if (delimiters.get(i).equals(openDel))
   {
      count++;
   }
   else
   {
      count--;
   }

   if (count < 0)
   {
      return false;
   }
}
```

```
    {
       return false;
    }
}
```

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

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

- Students need to practice comparing `String` values

Write program code to satisfy methods using expressions, conditional statements, and iterative statements.

- Students need to practice initializing and updating counters and accumulators.
- Students need to practice developing algorithms that test for multiple conditions during and after a traversal.

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

- Students need to practice constructing a new `ArrayList` of a specified type.
- Students need to practice traversing a one-dimensional array or `ArrayList` using an index.
- Students need to practice traversing a one-dimensional array or `ArrayList` using an enhanced `for` loop.

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

Suggested resources include:

Array/`ArrayList` free-response questions contained within the Personal Progress Checks in Units 6 and 7 provide two opportunities for students to practice writing program code to create, traverse, and manipulate elements in 1D array or `ArrayList` objects.

The Unit 6 Personal Progress Check, as well as the formative Topic Questions in Topic 6.2 and 6.3, provide students with practice traversing arrays using an index and an enhanced `for` loop, respectively. The Personal Progress Check in Unit 7 and the Topic Questions for Topics 7.1 – 7.4 provide practice creating, traversing and manipulating `ArrayList` objects.

For additional practice with these skills on summative items that are similar to this type of question, the AP Question Bank can be filtered to find questions that address skills 3.C and 3.D and focus on content from Units 6 and 7. However, because the Question Bank contains items from previous AP Exams and Official Practice Exams, these items should be previewed before assigning them to students to ensure that students are ready to apply all necessary skills and content knowledge that the items require.

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

- Data Lab is a College Board provided lab that contains a specific activity for students to practice manipulating elements in an `ArrayList`.
- The 2018 free-response question 2, Word Pair, requires students to create an `ArrayList` based on `String` values from a given array in part (a) and then traverse the `ArrayList` to find matching `String` values in part (b): https://secure-media.collegeboard.org/ap/pdf/ap18-frq-computer-science-a.pdf
- The current AP Computer Science A Course and Exam Description, Free-response sample Array/`ArrayList` question requires students to count elements of an `ArrayList` that match a given `String` in part (a):

https://apcentral-stg.collegeboard.org/pdf/ap-computer-science-a-course-and-exam-description.pdf?course=ap-computer-science-a

- The Practice-It! website hosted by the University of Washington offers practice for students to create, traverse, and manipulate elements in an `ArrayList`. This practice is consolidated into Chapter 10. This resource can be found here: https://practiceit.cs.washington.edu/problem/list
- The Runestone Interactive Java Review offers an interactive environment for students to practice course content. Specifically, the `List` and `ArrayList` sections would be most helpful for this question. This resource can be found here: http://interactivepython.org/runestone/static/JavaReview/index.html

Question #4 **Task:** 2D Array Processing **Topic:** Light Board

Max. Points: 9 **Mean Score:** 4.82

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 methods using expressions, conditional statements, and iterative statements; and
- Write program code to create, traverse, and manipulate elements in 2D array objects.

This question involved the creation and manipulation of a two-dimensional array of `boolean` values. Students were expected to implement a constructor and a method of the enclosing `LightBoard` class.

In part (a) students were asked to construct a two-dimensional array and initialize the values in the two-dimensional array based on a computed probability. Students were expected to be able to use the constructor's parameters to construct a two-dimensional `boolean` array with the correct number of rows and columns. Once the two-dimensional array was constructed, students were expected to write nested loops to access each item. For each item, the students were expected to use `Math.random()` to compute a probability for the purpose of choosing which `boolean` value should be assigned to the item.

In part (b) students were given parameters representing a `row` and `col` and were asked to evaluate the status of the light in the two-dimensional array at `lights[row][col]`. Students were expected to write a loop to access each item in the given column and use an accumulator to count the number of lights that are set to `true` in that column. Students were then expected to return a `boolean` value based on three rules: (1) If the light is on, return `false` if the number of lights in its column that are on is even, including the current light. (2) If the light is off, return `true` if the number of lights in its column that are on is divisible by three. (3) Otherwise, return the light's current status. To implement the rules, students were expected to perform an even calculation and a multiple of three calculation.

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.

Many responses struggled with generating a random number for this problem. This was especially true of students who tried to use an `int` instead of a `double` when calculating probability, since converting to an `int` requires using a cast appropriately.

Write program code to satisfy methods using expressions, conditional statements, and iterative statements.

While most responses were able to use a conditional statement to update a counter, some did not initialize the counter first. Some responses referenced Boolean values incorrectly in their expressions and others confused the requirements when writing compound Boolean expressions. Most responses were able to use the modulus operator to identify multiples of two and three.

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

Many responses could create a two-dimensional array of the correct size. However, responses struggled with specifying the correct type and assigning the created array to the instance variable `lights`.

Most responses were able to traverse all rows and columns of the array in part (a), but many had difficulty traversing only a single column in part (b).

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

Common Misconceptions/Knowledge Gaps *Write program code to create objects of a class and call methods.*	*Responses that Demonstrate Understanding*

Students failed to cast to an `int` when attemping to generate random integers within a range.
 	
`int rnd = Math.random() * 10;`
`lights[r][c] = rnd <= 3;`
 	
Students calculated a constant instead of a random number by leaving off necessary parentheses.
 	
`int rnd = (int) Math.random() * 10;`
`lights[r][c] = rnd <= 3;`
	`int rnd = (int)(Math.random() * 10);`
	`lights[r][c] = rnd < 4;`

Students used an incorrect range when attempting to calculate a 40% probability.
 	
`int rnd = (int)(Math.random() * 101);`
`lights[r][c] = rnd < 40;`
	`int rnd = (int)(Math.random() * 100);`
	`lights[r][c] = rnd < 40;`

Students used an incorrect comparison when attempting to calculate a 40% probability.
 	
`int rnd = (int)(Math.random() * 10);`
`lights[r][c] = rnd <= 4;`
	`int rnd = (int)(Math.random() * 10);`
	`lights[r][c] = rnd < 4;`

Common Misconceptions/Knowledge Gaps *Write program code to satisfy methods using expressions, conditional statements, and iterative statements.*	*Responses that Demonstrate Understanding*

Students reversed the sense of comparision when assigning `boolean` values.
 	
`for (int r = 0; r < numRows; r++)`
`{`
`   for (int c = 0; c < numCols; c++)`
`   {`
`      double rnd = Math.random();`
`      if (rnd > 0.4)`
`      {`
`         lights[r][c] = true;`
`      }`
`      else`
`      {`
`         lights[r][c] = false;`
`      }`
`   }`
	`for (int r = 0; r < numRows; r++)`
	`{`
	`   for (int c = 0; c < numCols; c++)`
	`   {`
	`      double rnd = Math.random();`
	`      if (rnd < 0.4)`
	`      {`
	`         lights[r][c] = true;`
	`      }`
	`      else`
	`      {`
	`         lights[r][c] = false;`
	`      }`
	`   }`

`}`	`}`

Students used a calculated probability of 50% which prevents determining a sense of comparison.
 	
`for (int r = 0; r < numRows; r++)`
`{`
`   for (int c = 0; c < numCols; c++)`
`   {`
`      int rnd =`
`         (int)(Math.random() * 10);`
`      if (rnd < 5)`
`      {`
`         lights[r][c] = true;`
`      }`
`      else`
`      {`
`         lights[r][c] = false;`
`      }`
`   }`
`}`
	`for (int r = 0; r < numRows; r++)`
	`{`
	`   for (int c = 0; c < numCols; c++)`
	`   {`
	`      int rnd =`
	`         (int)(Math.random() * 10);`
	`      if (rnd < 4)`
	`      {`
	`         lights[r][c] = true;`
	`      }`
	`      else`
	`      {`
	`         lights[r][c] = false;`
	`      }`
	`   }`
	`}`

Students did not initialize the counter variable. `int count;`	`int count = 0;`

Students used a division operator instead of the modulus operator when performing even or multiple of three calculations.
 	
`if (numOn / 2 == 0)…`
 	
Students wrote an incomplete `boolean` expression when performing an even or multiple of three calculation.
 	
`if (numOn % 2)…`
 	
Students performed a calculation involving modulus on a non-integer.
 	
`if (lights[row][col] % 2 == 0)…`
 	
Students compared two integers using `equals`.
 	
`if ((numOn % 2).equals(0))…`
	`if (numOn % 2 == 0)…`

Students called `equals` to check if two `boolean` values were equal.
 	
`if (lights[row][col].equals(true))…`
	`if (lights[row][col])…`
	`if (lights[row][col] == true)…`

Students misplaced conditionals inside the `for` loop causing an early return.
 	
`int numOn = 0;`
`for (int r = 0; r < lights.length; r++)`
	`int numOn = 0;`
	`for (int r = 0; r < lights.length; r++)`

`{`
`   if (lights[r][col])`
`   {`
`      numOn++;`
`   }`
 	
`   if (lights[row][col] && numOn % 2 == 0)`
`   {`
`      return false;`
`   }`
 	
`   if (!lights[row][col] &&`
`       numOn % 3 == 0)`
`   {`
`      return true;`
`   }`
`}`
`return lights[row][col];`
	`{`
	`   if (lights[r][col])`
	`   {`
	`      numOn++;`
	`   }`
	`}`
	
	`if (lights[row][col] && numOn % 2 == 0)`
	`{`
	`   return false;`
	`}`
	`if (!lights[row][col] && numOn % 3 == 0)`
	`{`
	`   return true;`
	`}`
	`return lights[row][col];`

Students coded a condition that never occurred.
 	
`if (lights[row][col])`
`{`
`   if (numOn % 2 == 0)`
`   {`
`      return false;`
`   }`
`}`
`else if (!lights[row][col])`
`{`
`   if (numOn % 3 == 0)`
`   {`
`      return true;`
`   }`
`}`
`else`
`{`
`   return lights[row][col];`
`}`
	`if (lights[row][col])`
	`{`
	`   if (numOn % 2 == 0)`
	`   {`
	`      return false;`
	`   }`
	`}`
	`else`
	`{`
	`   if (numOn % 3 == 0)`
	`   {`
	`      return true;`
	`   }`
	`}`
	
	`return lights[row][col];`

Students reversed the order of the even and multiple of three calculations.
 	
`if (lights[row][col] && numOn % 3 == 0)`
`{`
`   return false;`
`}`
`if (!lights[row][col] && numOn % 2 == 0)`
`{`
`   return true;`
`}`
	`if (lights[row][col] && numOn % 2 == 0)`
	`{`
	`   return false;`
	`}`
	`if (!lights[row][col] && numOn % 3 == 0)`
	`{`
	`   return true;`
	`}`

Common Misconceptions/Knowledge Gaps	*Responses that Demonstrate Understanding*
Write program code to create, traverse, and manipulate elements in 2D array objects.	

Students constructed the two-dimensional array with an incorrect data type.
 	
`lights = new int[numRows][numCols];`
 	
Students created a local two-dimensional array variable `lights`.
 	
`boolean[][] lights = new`
`             boolean[numRows][numCols];`
 	
Students omitted the keyword `new` when instantiating the two-dimensional array.
 	
`lights = boolean[numRows][numCols];`
	`lights = new boolean[numRows][numCols];`

Students created a local two-dimensional array without ever assigning it to the instance variable `lights`.
 	
`boolean[][] board = new`
`             boolean[numRows][numCols];`
	`boolean[][] board = new`
	`             boolean[numRows][numCols];`
	`lights = board;`

Students used incorrect bounds when accessing elements of the two-dimensional array `lights`.
 	
`for (int r = 0; r <= numRows; r++)`
`{`
`   for (int c = 0; c <= numCols; c++)`
`   {`
`      double rnd = Math.random();`
`      lights[r][c] = rnd < 0.4;`
`   }`
`}`
	`for (int r = 0; r < numRows; r++)`
	`{`
	`   for (int c = 0; c < numCols; c++)`
	`   {`
	`      double rnd = Math.random();`
	`      lights[r][c] = rnd < 0.4;`
	`   }`
	`}`

Students used incorrect bounds when accessing elements of a row in the two-dimensional array `lights`.
 	
`for (int r = 0; r < lights.length; r++)`
`{`
`   for (int c = 0; c < lights[].length;`
`        c++)`
`   {`
`      double rnd = Math.random();`
`      lights[r][c] = rnd < 0.4;`
`   }`
`}`
	`for (int r = 0; r < lights.length; r++)`
	`{`
	`   for (int c = 0; c < lights[0].length;`
	`        c++)`
	`   {`
	`      double rnd = Math.random();`
	`      lights[r][c] = rnd < 0.4;`
	`   }`
	`}`

Students incorrectly used an enhanced `for` loop for the inner loop to set the `boolean` values in the 2D array.
 	
`for (boolean[] row : lights)`
`{`
`   for (boolean r : row)`
`   {`
`      r = Math.random() < 0.4;`
`   }`
`}`
	`for (boolean[] row : lights)`
	`{`
	`   for (int c = 0; c < row.length; c++)`
	`   {`
	`      row[c] = Math.random() < 0.4;`
	`   }`
	`}`

Students stored an element of `lights` in a variable of type other than `boolean`. `int currentLight = lights[r][col];`	`boolean currentLight = lights[r][col];`

Students traversed a row instead of the specified `col`.
 	
`for (int c = 0; c < lights[0].length; c++)`
`{`
`   if (lights[row][c])`
`   …`
`}`
	`for (int r = 0; r < lights.length; r++)`
	`{`
	`   if (lights[r][col])`
	`   …`
	`}`

Students used incorrect bounds for the number of rows of the two-dimensonal array.
 	
`for (int r = 0; r < lights[0].length; r++)`
`{`
`   if (lights[r][col])`
`   …`
`}`
 	
`for (int r = 0; r < col; r++)`
`{`
`   if (lights[r][col])`
`   …`
`}`
	`for (int r = 0; r < lights.length; r++)`
	`{`
	`   if (lights[r][col])`
	`   …`
	`}`

Students did not attempt to count the number of lights in a column.
 	
`int numOn = 0;`
`if (lights[row][col] && numOn % 2 == 0)`
`{`
`   return false;`
`}`
`if (!lights[row][col] && numOn % 3 == 0)`
`{`
`   return true;`
`}`
`return lights[row][col];`
	`int numOn = 0;`
	`for (int r = 0; r < lights.length; r++)`
	`{`
	`   if (lights[r][col])`
	`   {`
	`      numOn++;`
	`   }`
	`}`
	
	`if (lights[row][col] && numOn % 2 == 0)`
	`{`
	`   return false;`
	`}`
	`if (!lights[row][col] && numOn % 3 == 0)`
	`{`
	`   return true;`
	`}`
	`return lights[row][col];`

Students implemented two counting loops but implemented one of them incorrectly.
 	
`int numOn = 0;`
`if (lights[row][col])`
`{`
`   for (int r = 0; r < lights.length; r++)`
`   {`
	`int numOn = 0;`
	`if (lights[row][col])`
	`{`
	`   for (int r = 0; r < lights.length; r++)`
	`   {`

```
      if (lights[r][col])
      {
         numOn++;
      }
   }
   return numOn % 2 != 0;
}
else
{
   for (int r = 0; r < lights[0].length;
        r++)
   {
      if (lights[r][col])
      {
         numOn++;
      }
   }
   return numOn % 3 == 0;
}
```

```
      if (lights[r][col])
      {
         numOn++;
      }
   }
   return numOn % 2 != 0;
}
else
{
   for (int r = 0; r < lights.length; r++)
   {
      if (lights[r][col])
      {
         numOn++;
      }
   }
   return numOn % 3 == 0;
}
```

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

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

- Students need to practice assigning created objects to instance variables.
- Students need to practice using `Math.random()` to compute a probability.

Write program code to satisfy methods using expressions, conditional statements, and iterative statements.

- Students need to practice initializing variables used as counters and accumulators.
- Students need to practice writing compound Boolean expressions.

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

- Students need to practice instantiating a new 2D array of a specified size and type.
- Students need to practice traversing all elements in a 2D array.
- Students need to practice traversing all elements in one column of a 2D array.
- Students need to practice assigning a value to an element of a 2D array based on some condition.

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

Suggested resources include:

Being able to combine the skills and content learned in Units 2-4 in the context of 2D array traversal and manipulation requires repeated exposure and practice. The Control Structures free-response questions that are included in the Personal Progress Checks in Units 2, 3, 4, and 10 provide several opportunities for students to practice writing program code to satisfy method specifications using expressions, conditional statements, and iterative statements. Unit 8 provides a Personal Progress Check that incorporates this content and skill with writing program code to create, traverse, and manipulate elements in 2D array objects. The formative Topic Questions in Topics 8.1 and 8.2 will provide additional exposure to 2D arrays.

For further practice with these skills on summative items that are similar to this type of question, the AP Question Bank can be filtered to find questions that address skills 3.C and 3.E and focus on content from Unit 8. However, because the Question Bank contains items from previous AP Exams and Official Practice Exams, these items should be previewed before assigning them to students to ensure that students are ready to apply all necessary skills and content knowledge that the items require.

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

- Picture Lab, Steganography Lab, and the GridWorld Curriculum Module provide students with opportunities to practice 2D array manipulation in the context of a larger programming project with multiple classes.
- The 2018 free-response question 4, Latin Squares, requires students to create a 2D array and access a single column: https://apcentral-stg.collegeboard.org/pdf/ap18-frq-computer-science-a.pdf
- The 2016 free-response question 3, Crossword, requires students to complete a constructor by instantiating a 2D array and assigning to an instance variable: https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap16_frq_computer_science_a.pdf
- The 2012 free-response question 4 requires students to implement a counter and assign values to elements in a 2D array: https://secure-media.collegeboard.org/apc/ap_frq_computerscience_12.pdf
- The Runestone Interactive Java Review offers an interactive environment for students to practice course content. Specifically, the Two-dimensional Arrays section would be most helpful for this question. This resource can be found here: http://interactivepython.org/runestone/static/JavaReview/index.html