Chief Reader Report on Student Responses:

2022 AP® Computer Science A Free-Response Questions

- Number of Students Scored 77,753
- Number of Readers 302
- Score Distribution

Exam Score	N	%At
5	21,196	27.3
4	15,843	20.4
3	15,476	19.9
2	8,072	10.4
1	17,166	22.1

- Global Mean 3.2

The following comments on the 2022 free-response questions for AP® Computer Science A were written by the Chief Reader, Alistair Campbell, Associate Professor of Computer Science at Hamilton College. 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: Video Game
Max Score: 9
Mean Score: 5.92

What were the responses to this question expected to demonstrate?

This question tested the student's ability to:

- Write program code to call methods.
- Write program code to satisfy method specifications using expressions, conditional statements, and iterative statements.

More specifically, this question assessed the ability to use `Level` objects, call methods within and outside the current class, use nested `if` logic to calculate a correct score for each level depending on whether that level and all previous levels' goals were met, iterate a specific number of times to identify a maximum score, and use method return values in conditional expressions.

In part (a) students were asked to declare and initialize a numeric score variable and then call the `getPoints` and `goalReached` methods from the `Level` class on the instance variables `levelOne`, `levelTwo`, and `levelThree` in order to calculate a correct score for each level, depending on whether that level and all previous levels' goals were met. Students then had to evaluate the returned value from `isBonus` to determine if the score for the game is tripled before being returned.

In part (b) students were asked to declare and initialize a maximum value variable, iterate `num` times to call the `play` method, and compare the value returned from `getScore` to the identified maximum, replacing the maximum as needed in the loop after a correct comparison. The students then had to return the identified maximum score.

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 call methods.

In part (a) responses called two methods of the `Level` class and one method of the `Game` class. In particular, the methods from the `Level` class, `getPoints` and `goalReached`, are called on the instance variables `levelOne`, `levelTwo`, and `levelThree`. The method `isBonus` is an instance method of `Game` and could be called without a qualifier in the `getScore` method.

In part (b) responses called the no-parameter methods `play` and `getScore`. These two methods are in the `Game` class and are being called from within a third `Game` method, so no qualifier is needed when calling them. Many responses successfully called the methods with no parameters and then used the returned results appropriately.

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

In part (a) responses declared and initialized a numeric score variable. Then they called the required methods using nested `if` statements or equivalent logic to calculate a correct score for each of three levels, depending on whether that level and all previous levels' goals were met. Finally, they wrote a conditional expression to possibly triple the score value before returning it. Most responses created the score variable correctly. Many responses used correct logic to calculate a score for two of the three levels but not for all three levels. Most responses correctly used a conditional expression to only triple the score variable when applicable, but some responses incorrectly tripled the score by just calculating the new score and then not assigning the calculated value to the score variable. Returning was not assessed in this part of the question. The most common issue was not calculating a correct score for each level.

In part (b) responses declared and initialized an identified maximum value variable and then iterated `num` times to determine the maximum value. Within the loop, they called the required methods and wrote a comparison statement to update the maximum variable. After the loop, they returned this value. The majority of responses successfully compared the score values and created/returned an identified maximum value.

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

Common Misconceptions/Knowledge Gaps *Write program code to call methods.*	*Responses that Demonstrate Understanding*

Some responses treated `isBonus` as a static method.
 	
`if (Game.isBonus())`
or
`if (Level.isBonus())`
	`if (isBonus())`
	or
	`if (this.isBonus())`

Some responses combined the `play` and `getScore` method calls. `int score = play().getScore();`	`play();` `int score = getScore();`

Some responses failed to call `getPoints` and mistakenly called `getScore`, which is the method being written.
 	
`if (levelOne.goalReached())`
`{`
`  score = levelOne.getScore();`
`}`
	`if (levelOne.goalReached())`
	`{`
	`  score = levelOne.getPoints();`
	`}`

Some responses omitted the method call to `getPoints` and assigned numeric values instead.
 	
`if (levelOne.goalReached())`
`{`
`  score += 200;`
`}`
	`if (levelOne.goalReached())`
	`{`
	`  score = levelOne.getScore();`
	`}`

Some responses called `play` and `getScore` using `Game` as the qualifier or with parameters.
 	
`Game.play()`
or
`Game.getScore()`
or
`play(num)`
	`play()`
	`getScore()`

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 calculate a correct score for each of the three levels depending on whether that level's and all previous levels' goals were met.
 	
`if (levelOne.goalReached())`
`{`
`  score += levelOne.getPoints();`
`}`
`if (levelTwo.goalReached())`
`{`
`  score += levelTwo.getPoints();`
`}`
`if (levelThree.goalReached())`
`{`
`  score += levelThree.getPoints();`
`}`
or
`if (levelOne.goalReached())`
`{`
`  score += levelOne.getPoints();`
`}`
`else if (levelOne.goalReached() &&`
`  levelTwo.goalReached())`
`{`
`  score += levelTwo.getPoints();`
`}`
`else if (levelOne.goalReached() &&`
`  levelTwo.goalReached() &&`
`  levelThree.goalReached())`
`{`
`  score += levelThree.getPoints();`
`}`
	`if (levelOne.goalReached())`
	`{`
	`  score += levelOne.getPoints();`
	`  if (levelTwo.goalReached())`
	`  {`
	`  score += levelTwo.getPoints();`
	`  if (levelThree.goalReached())`
	`  {`
	`  score += levelThree.getPoints();`
	`  }`
	`  }`
	`}`
	or
	`if (levelOne.goalReached())`
	`{`
	`  score += levelOne.getPoints();`
	`}`
	`if (levelOne.goalReached() &&`
	`  levelTwo.goalReached())`
	`{`
	`  score += levelTwo.getPoints();`
	`}`
	`if (levelOne.goalReached() &&`
	`  levelTwo.goalReached() &&`
	`  levelThree.goalReached())`
	`{`
	`  score += levelThree.getPoints();`
	`}`

Some responses tripled the score only when **level three** was reached.

```
if (levelOne.goalReached())
{
   score += levelOne.getPoints();
   if (levelTwo.goalReached())
   {
      score += levelTwo.getPoints();
      if (levelThree.goalReached())
      {
         score += levelThree.getPoints();
         if (isBonus())
         {
            score *= 3;
         }
      }
   }
}
```

```
if (levelOne.goalReached())
{
   score += levelOne.getPoints();
   if (levelTwo.goalReached())
   {
      score += levelTwo.getPoints();
      if (levelThree.goalReached())
      {
        score += levelThree.getPoints();
      }
   }
}
if (isBonus())
{
   score *= 3;
}
```

Many responses used the correct logic to calculate a score for two of the three levels but not for all three levels.

```
if (!levelOne.goalReached())
{
   return 0;
}
else
{
   score += levelOne.getPoints();
}
if (levelTwo.goalReached())
{
   score += levelTwo.getPoints();
}
if (levelThree.goalReached())
{
   score += levelThree.getPoints();
}
```

```
int mult = 1;
if (isBonus())
{
   mult = 3;
}

if (!levelOne.goalReached())
{
   return 0;
}
else
{
   score += levelOne.getPoints();
}
if (!levelTwo.goalReached())
{
   return mult * score;
}
else
{
   score += levelTwo.getPoints();
}

if (!levelThree.goalReached())
{
   return mult * score;
}
else
{
   score += levelThree.getPoints();
}
return mult * score;
```

Some responses did not initialize the maximum variable.
 	
`int max;`
`…`
`return max;`
	`int max = 0;`
	`…`
	`return max;`

Some responses did not iterate the specified number of times.
 	
`for (int i = 0; i <= num; i++)`
or
`for (int i = 1; i < num; i++)`
or
`for (int i = 0; i < num - 1; i++)`
	`for (int i = 0; i < num; i++)`
	
	`for (int i = 1; i <= num; i++)`
	
	`for (int i = 0; i <= num - 1; i++)`

Some responses failed to call `play` exactly once each time through the loop.
 	
`play();`
`for (int i = 0; i < num; i++)`
`{`
`  if (getScore() > max)`
`  {`
`  max = getScore();`
`  }`
`}`
	`for (int i = 0; i < num; i++)`
	`{`
	`  play();`
	`  if (getScore() > max)`
	`  {`
	`  max = getScore();`
	`  }`
	`}`

Some responses incorrectly tripled the score by not updating the score variable.
 	
`if (isBonus())`
`{`
`  score * 3;`
`}`
	`if (isBonus())`
	`{`
	`  score *= 3;`
	`}`

Some responses calculated a sum or an average instead of determining a maximum value.
 	
`for (int i = 0; i < num; i++)`
`{`
`  play();`
`  max += getScore();`
`}`
`return max / num;`
	`for (int i = 0; i < num; i++)`
	`{`
	`  play();`
	`  if (getScore() > max)`
	`  {`
	`  max = getScore();`
	`  }`
	`}`
	`return max;`

Some responses used a two-loop solution to determine the maximum, but made errors in creating, traversing, or manipulating their temporary array or `ArrayList`.

```
int[] scoreList = [num];

for (int i = 0; i < num; i++)
{
   play();
   scoreList[i] = getScore();
}

int temp = 0;
for (int j = 0; j < scoreList.size; j++)
{
   if (scoreList[j] > temp)
   {
      temp = scoreList[j];
   }
}
return temp;
```

or

```
ArrayList<Integer> scores =
   ArrayList<int>();

for (int j = 0; j < num; j++)
{
   play();
   scores.set(j, getScore());
}

int temp = 0;
for (int j = 0; j < scores.size; j++)
{
   if (scores[j] > temp)
   {
      temp = scores[j];
   }
}
return temp;
```

```
int[] scoreList = new int[num];

for (int i = 0; i < num; i++)
{
   play();
   scoreList[i] = getScore();
}

int temp = 0;
for (int j = 0; j < scoreList.length;
     j++)
{
   if (scoreList[j] > temp)
   {
      temp = scoreList[j];
   }
}
return temp;
```

or

```
ArrayList<Integer> scores = new
   ArrayList<Integer>();

for (int j = 0; j < num; j++)
{
   play();
   scores.add(getScore());
}

int temp = 0;
for (int j = 0; j < scores.size(); j++)
{
   if (scores.get(j) > temp)
   {
      temp = scores.get(j);
   }
}
return temp;
```

Some responses created a new `Game` object and used it to call the required methods.

```
Game g = new Game();
for (int i = 0; i < num; i++)
{
   g.play();
   if (g.getScore() > max)
   {
      max = g.getScore();
   }
}
```

```
for (int i = 0; i < num; i++)
{
   play();
   if (getScore() > max)
   {
      max = getScore();
   }
}
```

Some responses had an early return from the loop after comparing two scores.
 	
`for (int i = 0; i < num; i++)`
`{`
`  play();`
`  if (getScore () > max)`
`  {`
`  return getScore();`
`  }`
`  else`
`  {`
`  return max;`
`  }`
`}`
	`for (int i = 0; i < num; i++)`
	`{`
	`  play();`
	`  if (getScore() > max)`
	`  {`
	`  max = getScore();`
	`  }`
	`}`
	`return max;`

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?

- Reinforce the relationship between an object and a method of the object's class
- Reinforce the difference between a static method and an instance method
- Continue practicing loop logic to repeat a specific number of times
- Continue practicing logic to determine a max value
- Reinforce the concept of curly brackets creating a block of code to prevent early returns and incorrect nesting

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: Textbook
Max Score: 9
Mean Score: 5.03

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 call methods.
- Write program code to satisfy method specifications using expressions and conditional statements.

Students were given a class, `Book`, and asked to design a subclass called `Textbook`. The `Book` class contained the title and the price of the book and accessor methods for this information. In implementing a solution, students were expected to demonstrate an understanding of class constructor and method header syntax. Students were expected to properly declare and initialize a new private instance variable to maintain the edition number. Further, students had to recognize that the `title` and `price` variables in the `Book` class were private and could not be accessed directly by the code. Students had to utilize the mechanisms of inheritance to initialize and access these variables by way of a `super` constructor call from the `Textbook` class to the `Book` class.

The specification for the `Textbook` class required two new methods be added: `getEdition` to return the edition of the textbook and `canSubstituteFor` to check if the textbook could be substituted for a given textbook. In the `canSubstituteFor` method, students were expected to compare the titles of two `Textbook` objects via the `equals` method of the `String` class. Students were also expected to use arithmetic relational operators and combine the results of multiple comparisons via Boolean logic or conditional statements.

Additionally, students were required to override the `getBookInfo` method by calling the superclass method, and to demonstrate the ability to construct a `String` containing multiple pieces of information.

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?

Successful class designs created an instance variable to store the edition number and initialized it in the constructor. Further, a successful design utilized inheritance to handle title and price information by calling the superclass's constructor and utilizing the `getTitle` and `getBookInfo` methods of the `Book` class.

While many responses used this strategy, they often made mistakes with the details or did not fully utilize inheritance.

Many responses called the `super` constructor appropriately, but then directly accessed the private instance variables `title` and `price` rather than using the appropriate methods. Other responses declared new instance variables for title and price information, initializing them in the constructor, and then called the methods of the superclass. Both of these incomplete uses of inheritance mechanisms indicate a lack of understanding of the details of class design using inheritance.

There were also a number of responses where students did not utilize inheritance at all and re-implemented the details of the `Book` class.

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

Common Misconceptions/Knowledge Gaps *Write code to define a new type by creating a class.*	*Responses that Demonstrate Understanding*

Some responses did not use inheritance.
 	
`public class Textbook`
 	
Some responses provided parameters for the class header.
 	
`public class Textbook extends Book(String title, double price, int edition)`
	`public class Textbook extends Book`

Some responses provided parameters in the wrong order.
 	
`public Textbook(int editionNumber, String title, double price)`
 	
Some responses omitted the parameters for title and price information.
 	
`public Textbook(int editionNumber)`
 	
Some responses omitted parameters.
 	
`public Textbook()`
 	
Some responses had the wrong types for parameters.
 	
`public Textbook(String title, String price, String editionNumber)`
 	
Some responses omitted the constructor.
	`public Textbook(String title, double price, int editionNumber)`

Some responses declared local variables inside the constructor instead of instance variables.

```
public Textbook(String title,
   double price, int editionNumber)
{
   super(title, price);
   int edition = editionNumber;
}
```

Some responses reversed the instance variable and the parameter in the assignment statement.

```
private int edition;

public Textbook(String title,
   double price, int editionNumber)
{
   super(title, price);
   editionNumber = edition;
}
```

```
private int edition;

public Textbook(String title, double price,
   int editionNumber)
{
   super(title, price);
   edition = editionNumber;
}
```

Some responses declared new instance variables for the title and price information.

```
private String title;
private double price;
private int edition;

public Textbook(String t, double p,
   int e)
{
   title = t;
   price = p;
   edition = e;
}
```

```
private int edition;

public Textbook(String title, double price,
   int editionNumber)
{
   super(title, price);
   edition = editionNumber;
}
```

Some responses omitted one or more of the methods `getEdition`, `canSubstituteFor`, and `getBookInfo`.
 	
Some responses added parameters to the methods where none were specified.
 	
`public int getEdition(int ed)`
 	
`public String getBookInfo(String title, double price)`
 	
Some responses had the wrong type of parameters.
 	
`public boolean canSubstituteFor(String other)`
 	
`public boolean canSubstituteFor(String title, double price)`
 	
Some responses failed to provide a return type.
 	
`public getEdition()`
 	
Some responses attempted to use an additional parameter rather than using the information in the object's instance variables.
 	
`public boolean canSubstituteFor(Textbook objectTextbook, Textbook other)`
	`public boolean canSubstituteFor(Textbook other)`
	
	`public int getEdition()`
	
	`public String getBookInfo()`

Common Misconceptions/Knowledge Gaps *Write program code to call methods.*	*Responses that Demonstrate Understanding*

Some responses did not call the superclass's constructor correctly.
 	
`public Textbook(String title, double price, int editionNumber) { super(title); super(price); edition = editionNumber; }`
 	
Some responses did not call `super` as the first line of the constructor.
 	
`public Textbook(String title, double price, int editionNumber) { edition = editionNumber; super(title, price); }`
	`private int edition;`
	
	`public Textbook(String title, double price, int editionNumber) { super(title, price); edition = editionNumber; }`

Some responses directly accessed the private instance variables `title` and `price`.
 	
`title.equals(other.title)`
 	
`return title + "-" + price + "-" + getEdition();`
 	
Some responses assumed the existence of a method `getPrice` in the `Book` class.
 	
`return getTitle() + "-" + getPrice() + "-" + getEdition();`
	`getTitle().equals(other.getTitle())`
	
	`return super.getBookInfo() + "-" + getEdition();`

Some responses attempted to use a method by specifying the class name rather than the name of an instance of the class.
 	
`Textbook.getTitle()`
	`getTitle()`
	
	`super.getBookInfo()`
	
	`other.getTitle()`

Common Misconceptions/Knowledge Gaps *Write code to satisfy method specifications using expressions and conditional statements.*	*Responses that Demonstrate Understanding*
Some responses used `==` to compare strings for equality. `getTitle() == other.getTitle()`	`getTitle().equals(other.getTitle())`

Some responses used an incorrect inequality when comparing edition numbers.
 	
`getEdition() > other.getEdition()`
or
`getEdition() <= other.getEdition()`
	`getEdition() >= other.getEdition()`

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 5 and 9 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 units 5 and 9.
  - *Write program code to create objects of a class and call methods.* Topics 2.3, 2.4, 2.5, 9.4, and 9.6.
  - *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: Review Analysis
Max Score: 9
Mean Score: 4.48

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 and `ArrayList` objects.

This question involved the traversal and manipulation of a one-dimensional (1D) array containing `Review` objects and the instantiation and building of an `ArrayList` containing `String` objects. Students were expected to write two methods in the `ReviewAnalysis` class, using its 1D array instance variable, and use two methods from the `Review` class when accessing `Review` objects. Students were also expected to use methods of the `String` class and construct a `String` object.

In part (a) students were expected to write a loop that accessed each element of the array instance variable `allReviews` and returned the calculated average of all the return values of the method `getRating`. Students had to declare and initialize a variable to hold the sum of all ratings. Inside the loop, students were expected to call the method `getRating` on all `Review` elements in `allReviews`, accumulating a total of the ratings. Outside the loop, students were expected to calculate and return the average rating as a `double` value.

In part (b) students were asked to develop an algorithm to: (1) Identify all `Review` elements of `allReviews` that have comments containing an exclamation point, using the `getComment` method; (2) build an `ArrayList` of `String` objects, each of which would be a string based on an identified review and formatted according to the specification:

> "<index number of `Review` object><hyphen><comment>"

When building the formatted `String` to be added to the `ArrayList`, the specification required that each identified formatted comment end in a period if the original comment did not already end with a period or an exclamation point. In identifying the comments meeting the specification, students were to use methods of the `String` class, such as `indexOf`, `substring`, and `equals`, as well as Boolean operators.

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) responses generally accessed all elements of `allReviews` properly, without off-by-one bounds errors. Responses also called a no-argument method properly on a `Review` object and accumulated the

return values appropriately. In part (b) responses usually created an `ArrayList` containing `String` objects and used the `add` method appropriately to populate it.

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

In part (a) responses used iteration to form the arithmetic sum of all ratings. In part (b) responses successfully iterated over all comments and usually used some sort of conditional expression to decide which comments should be included in the `ArrayList`. Most responses properly constructed the hyphenated `String` using `String` concatenation expressions.

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

In part (a) responses iterated over the `allReviews` array using either an indexing `for` loop or an enhanced `for` loop. In part (b) responses were generally successful at keeping track of the loop index so the hyphenated string could be constructed. The constructed `ArrayList` tended to be of the correct type and accessed properly.

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*

Some responses failed to call `getRating` on a `Review` element.
 	
`for (int i = 0; i < allReviews.length; i++)`
`{`
`  sum += allReviews[i];`
`}`
	`for (int i = 0; i < allReviews.length; i++)`
	`{`
	`  sum += allReviews[i].getRating();`
	`}`

Some responses failed to correctly call the `indexOf` method of the `String` class.
 	
`String comment =`
`  allReviews[i].getComment();`
`  if (comment.indexAt("!") > -1)`
`  …`
or
`String comment =`
`  allReviews[i].getComment();`
`  if (comment.getIndexAt("!") > -1)`
`  …`
	`String comment =`
	`  allReviews[i].getComment();`
	`  if (comment.indexOf("!") > -1)`
	`  …`

Many responses treated each element of the `allReviews` array as a `String` object.
 	
`for (int i = 0; i < allReviews.length; i++)`
`{`
`  String comment = allReviews[i];`
`}`
or
`for (int i = 0; i < allReviews.length; i++)`
`{`
`  if (allReviews[i].indexOf("!") >= 0)`
`  …`
	`for (int i = 0; i < allReviews.length; i++)`
	`{`
	`  String comment =`
	`  allReviews[i].getComment();`
	`  …`
	or
	`for (int i = 0; i < allReviews.length; i++)`
	`{`
	`  if (allReviews[i].getComment().indexOf("!") >= 0)`
	`  …`

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 calculate and return the average as a `double` when using an `int` accumulator.
 	
`int sum = 0;`
`…`
`return sum / allReviews.length;`
	`int sum = 0;`
	`…`
	`return (double)sum / allReviews.length;`

Some responses attempted to compare `String` objects using `==` or `!=` operators.
 	
`String last =`
`  comment.substring(comment.length - 1);`
`  if (last != "!" && last != ".")`
`  …`
	`String last =`
	`  comment.substring(comment.length - 1);`
	`  if(!last.equals("!") &&`
	`  !last.equals("."))`
	`  …`

Some responses used incorrect logic to determine if a period should be added at the end of the comment.

```
if (last.equals("!") || last.equals("."))
{
   result.add(i + "-" + comment + ".");
}
else
{
   result.add(i + "-" + comment);
}
```

```
if (!last.equals("!") && !last.equals("."))
{
   result.add(i + "-" + comment + ".");
}
else
{
   result.add(i + "-" + comment);
}
```

or

```
if (!(last.equals("!") ||
      last.equals(".")))
{
   result.add(i + "-" + comment + ".");
}
else
{
   result.add(i + "-" + comment);
}
```

or

```
if (last.equals("!") || last.equals("."))
{
   result.add(i + "-" + comment);
}
else
{
   result.add(i + "-" + comment + ".");
}
```

Some responses failed to compare final characters at all.

```
for (…)
{
   String com = allReviews[i].getComment();
   if (com.indexOf("!") != -1)
   {
      result.add(i + "-" + com);
   }
}
```

```
for (…)
{
   String com = allReviews[i].getComment();
   if (com.indexOf("!") != -1)
   {
      if (…)
      {
         result.add(i + "-" + com + ".");
      }
      else
      {
         result.add(i +  "-" + com);
      }
   }
}
```

Many responses failed to keep track of the index.

```
for (Review r : allReviews)
{
   String com = r.getComment();
   if (…)
   {
      String result = r.getRating() +
         "-" + com;
      …
```

```
for (int i = 0; i < allReviews.length; i++)
{
   String com = allReviews[i].getComment();
   if (…)
   {
      String result = i + "-" + com;
      …
```

Common Misconceptions/Knowledge Gaps

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

Responses that Demonstrate Understanding

Some responses failed to access all elements of the `allReviews` array.

```
for (int i = 0; i < allReviews.length - 1;
     i++)
{
   sum += allReviews[i].getRating();
}
```

or

```
for (int i = 0; i <= allReviews.length;
     i++)
{
   sum += allReviews[i].getRating();
}
```

```
for (int i = 0; i < allReviews.length; i++)
{
   sum += allReviews[i].getRating();
}
```

or

```
for (int i = 0; i <= allReviews.length - 1;
     i++)
{
   sum += allReviews[i].getRating();
}
```

Some responses accessed elements of the `allReviews` array as if from an `ArrayList`.

```
for (int i = 0; i < allReviews.length; i++)
{
   String comment =
      allReviews.get(i).getComment();
   …
```

```
for (int i = 0; i < allReviews.length; i++)
{
   String comment =
      allReviews[i].getComment();
   …
```

Some responses failed to correctly instantiate an `ArrayList` of `String` objects.

```
ArrayList<> result =
   new ArrayList<String>();
```

or

```
ArrayList<String> result =
   ArrayList<String>();
```

or

```
String[] result = new String[];
```

```
ArrayList<String> result =
   new ArrayList<String>();
```

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

- Review the difference between `double` division and `int` division (Topics 1.2 and 1.3).
- Review of `String` class methods (Topic 2.7), especially using the `equals` method to compare `String` objects.
- Review compound Boolean expressions, especially De Morgan's Laws (Topics 3.5 and 3.6).
- Personal progress checks from units 6 and 7 would be helpful to scaffold students' understanding for the Array / `ArrayList` free-response questions.
- The following AP Daily Videos and corresponding Topic Questions can be found in AP Classroom to support this Array / `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 array and `ArrayList` objects.* Topics 6.1, 6.2, 7.2, 7.3, 7.4, and 7.5.

Question 4

Task: 2D Array
Topic: Random Grid
Max Score: 9
Mean Score: 3.86

What were the responses to this question expected to demonstrate?

This question tested the student's ability to:

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

This question involved the manipulation of a two-dimensional (2D) array of `int` values. A class that included two methods, one written in part (a) and one written in part (b), was provided.

In part (a) students were asked to write a `void` method, `repopulate`, that assigned newly generated random numbers to each element of the 2D array instance variable, `grid`. The new elements in the array must be between 1 and the class constant `MAX`, inclusive; must be divisible by 10; and must not be divisible by 100. All values must have an equal chance of being generated. Students were expected to traverse all elements of the array and assign a newly generated random number satisfying each of the criteria to each array element.

In part (b) students were asked to write a method, `countIncreasingCols`, that returned the number of columns in the 2D array instance variable, `grid`, that are in increasing order. A column is in increasing order when each element of the column after the first row is greater than or equal to the element of the column in the previous row. Students were expected to traverse the array in column-major order. Students were expected to identify columns in which each pair of adjacent elements satisfy the increasing criterion. Students were then expected to count the identified columns and return the count.

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 call methods.

Most responses were able to call the methods needed in this problem. However, there were some that did not use the appropriate arguments to call the methods.

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

Most responses were able to call `Math.random()` correctly, but many had difficulty scaling the resulting random number correctly (by multiplying by `MAX`), translating the values to the range `[1, MAX]` (by adding 1), or converting the value to an `int` using a cast. While most responses were able to use conditional

expressions, often these conditions had one or more errors. While most responses were able to form comparisons between expressions, some did not correctly form accesses to two elements in the same column, or used the wrong comparison operator. Many responses were able to form a condition to identify an increasing column, but some response expressions contained off-by-one errors, or did not give the counter variable an initial value.

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

Many responses could traverse a 2D array in part (a) but could not correctly form a column-major traversal 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 call methods.*	*Responses that Demonstrate Understanding*
Some responses called parameter-less methods with parameters. `Math.random(1, MAX)`	`Math.random()`
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 generate a random integer in the range [`1, MAX`].
 	
`int rval = (int)Math.random() + 1;`
or
`int rval = (int)(Math.random() * MAX);`
or
`int rval = (Math.random() * MAX) + 1;`
or
`int rval = (int)(Math.random() * MAX) + 10;`
or
`int rval = (int)(Math.random() * 100) + 1;`
	`int rval = (int)(Math.random() * MAX) + 1;`

Some responses failed to ensure that all produced values are divisible by 10 but not by 100.
 	
`while (rval % 10 != 0 && rval % 100 == 0)`
`{`
`  rval = (int)(Math.random() * MAX) + 1;`
`}`
or
`while (rval % 10 == 0 || rval % 100 != 0)`
`{`
`  rval = (int)(Math.random() * MAX) + 1;`
`}`
or
`if (rval % 10 == 0 || rval % 100 != 0)`
`{`
`  grid[row][col] = rval;`
`}`
or
`if (rval % 10 != 0 && rval % 100 == 0)`
`{`
`  grid[row][col] = rval;`
`}`
	`while (rval % 10 != 0 || rval % 100 == 0)`
	`{`
	`  rval = (int)(Math.random() * MAX) + 1;`
	`}`

Some responses failed to ensure that all appropriate random values have an equal chance of being generated.
 	
`int rval = (int)Math.random() * MAX + 1;`
or
`int rval =`
`  (int)(Math.random() * MAX / 10) + 1;`
or
`int rval = (int)(Math.random() * MAX) + 1;`
`if (rval % 10 != 0)`
`{`
`  rval -= rval % 10;`
`}`
`if (rval % 100 == 0)`
`{`
`  rval -= 10;`
`}`
	`int rval = (int)(Math.random() * MAX) + 1;`

Some responses failed to correctly compare two elements of the same column.
 	
`if (grid[row][col] < grid[row][col - 1])`
or
`if (grid[row][col] <`
`  grid[row - 1][col - 1])`
or
`if (grid[row][col] > grid[row - 1][col])`
or
`if (grid[row][col] <= grid[row - 1][col])`
	`if (grid[row][col] < grid[row - 1][col])`
	or
	`int prev = grid[0][col];`
	`for (int row = 1; row < grid.length;`
	`  row++)`
	`{`
	`  if (grid[row][col] < prev)`
	`  {`
	`  ordered = false;`
	`  }`
	`  prev = grid[row][col];`
	`}`

Some responses failed to correctly identify an increasing column.

```
int ordered = 0;
for (int row = 1; row < grid.length; row++)
{
   if (grid[row][col] < grid[row - 1][col])
   {
      ordered++;
   }
}
if (ordered == grid.length)
{
   count++;
}
```

or

```
int count = 0;
for (int col = 0; col < grid[0].length;
     col++)
{
   for (int row = 1; row < grid.length;
        row++)
   {
      if (grid[row][col] <
          grid[row - 1][col])
      {
         count++;
      }
   }
}
```

```
int ordered = 0;
for (int row = 1; row < grid.length;
     row++)
{
  if (grid[row][col] < grid[row - 1][col])
  {
     ordered++;
  }
}
if (ordered == grid.length - 1)
{
   count++;
}
```

Some responses failed to reset variables in the outer loop before processing the next column.
 	
`for (int col = 0; col < grid[0].length; col++)`
`{`
`  for (int row = 1; row < grid.length; row++)`
`  {`
`  …`
 	
or
 	
`boolean ordered = true;`
`for (int col = 0; col < grid[0].length; col++)`
`{`
`  for (int row = 1; row < grid.length; row++)`
`  {`
`  …`
 	
or
 	
`for (int col = 0; col < grid[0].length; col++)`
`{`
`  for (int row = 1; row < grid.length; row++)`
`  {`
`  boolean ordered = true;`
`  …`
	`for (int col = 0; col < grid[0].length; col++)`
	`{`
	`  boolean ordered = true;`
	`  for (int row = 1; row < grid.length; row++)`
	`  {`
	`  …`

Some responses failed to give the column counter variable an initial value. `int count;` or had no declaration of the counter variable.	`int count = 0;`
Common Misconceptions/Knowledge Gaps *Write program code to create, traverse, and manipulate elements in 2D array objects.*	*Responses that Demonstrate Understanding*
Some responses compared entire rows of a 2D array instead of individual elements. `if (grid[row] < grid[row - 1])`	`if (grid[row][col] < grid[row - 1][col])`

Some responses failed to use integer indices in array references when using enhanced `for` loops.

```
for (int[] row : grid)
{
   for (int col : row)
   {
      …
      grid[row][col] = rval;
   }
}
```

```
int i = 0;
for (int[] row : grid)
{
   int j = 0;
   for (int col : row)
   {
      …
      grid[i][j] = rval;
      j++;
   }
   i++;
```

Some responses used incorrect bounds when accessing elements of the 2D array `grid`.

```
for (int r = 0; r < grid.length; r++)
{
   for (int c = 0; c < grid.length; c++)
   {
      …
      grid[r][c] = rval;
   }
}
```

or

```
for (int r = 0; r < grid.length - 1; r++)
{
   for (int c = 0; c < grid[0].length; c++)
   {
      …
      grid[r][c] = rval;
   }
}
```

or

```
for (int c = 0; c < grid[0].length; c++)
{
   for (int r = 0; r < grid.length; r++)
   {
      if (grid[r][c] < grid[r + 1][c])
      …
   }
}
```

```
for (int r = 0; r < grid.length; r++)
{
   for (int c = 0; c < grid[0].length; c++)
   {
      …
      grid[r][c] = rval;
   }
}
```

or

```
for (int c = 0; c < grid[0].length; c++)
{
   for (int r = 0; r < grid.length - 1;
        r++)
   {
      if (grid[r][c] < grid[r + 1][c])
      …
   }
}
```

Some responses failed to traverse the 2D array `grid` in column-major order.
 	
`for (int r = 1; r < grid.length; r++)`
`{`
`  for (int c = 0; c < grid[0].length; c++)`
`  {`
`  …`
`  }`
`}`
	`for (int c = 0; c < grid[0].length; c++)`
	`{`
	`  for (int r = 1; r < grid.length; r++)`
	`  {`
	`  …`
	`  }`
	`}`

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 2.3, 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, 3.5, 4.2, and 4.4.
    Write program code to create, traverse, and manipulate elements in 2D array objects. Topics 8.1 and 8.2.