Chief Reader Report on Student Responses:

2021 AP® Computer Science A Free-Response Questions

• Number of Students Scored	74,676		
• Number of Readers	413		
• Score Distribution	Exam Score	N	%At
	5	17,845	23.9
	4	16,348	21.9
	3	14,392	19.3
	2	9,047	12.1
	1	17,044	22.8
• Global Mean	3.12		

The following comments on the 2021 free-response questions for AP® Computer Science A were written by the Chief Reader, John Cigas, Professor of Computer Science at 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:** Secret Strings

Max. Points: 9 **Mean Score:** 4.57

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.

More specifically, this question assessed the ability to use `String` objects, iterate through a range, call `String` methods, and use a method's return value in a conditional expression.

In part (a) students were asked to loop through substrings of `secret` to determine whether there is an occurrence of the string `guess` within `secret`. Students accumulated a count of the number of occurrences of `guess` within `secret`. They were expected to initialize a numeric counter, iterate through all the substrings of `secret`, and update the counter. The students then had to calculate the return value, which is the product of their counter and the square of the length of `guess`.

In part (b) students were asked to compare the results of a method call using conditional statements. They needed to test which return value from two calls to `scoreGuess` was greater and return the parameter with the higher return value. The students also needed to perform an alphabetical comparison of the two parameters if the return values from the `scoreGuess` method calls were equal. They needed to return the correct string based on their comparisons.

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 called methods of the `String` class. In particular, responses needed to use `length`, `equals`, `indexOf`, and `substring`. Some responses failed to use `equals,` or `compareTo,` when comparing strings. Many responses failed to compare all necessary substrings. The most common problems were failing to access substrings of the correct length, skipping some substrings, and accessing positions not in the original string.

In part (b) responses called the `WordMatch` method `scoreGuess` twice, once with `guess1` as the sole parameter, and again with `guess2` as the parameter. The parameters to the `scoreGuess` method were themselves the parameters passed into the `findBetterGuess` 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) responses iterated over multiple substrings of the `secret` instance variable. Within the loop, they wrote a conditional expression to update a counter. After the loop, they computed the product of the computed value and the square of the parameter's length and returned this value. Many students did this correctly, but had difficulties enumerating all necessary substrings correctly. The most common issues were exceeding the bounds of the subject string and skipping over some substrings.

In part (b) responses used the result of three method calls to perform a calculation, with the final result being one of the two parameters to the method. The majority of students successfully called the methods and returned the correct value for some, but not all cases. Many students had difficulty comparing two strings alphabetically and were therefore unable to return the correct result for the alphabetical case.

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 treated `scoreGuess` as a method on a string instead of on a `WordMatch` object.
`if (guess1.scoreGuess() >`
`                       guess2.scoreGuess())`
	`if (scoreGuess(guess1) >`
	`                   scoreGuess(guess2))`

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

Some responses failed to iterate over all necessary substrings of `secret`.
`for (int i = 0; i < secret.length() –`
`                       guess.length(); i++)`
`{`
`  if (secret.substring(i, i +`
`             guess.length()).equals(guess))`
`  …`
Some responses exceeded the bounds of the string `secret`.
`for (int i = 0; i < secret.length(); i++)`
`{`
`  if (secret.substring(i, i +`
`             guess.length()).equals(guess))`
`  …`
	`for (int i = 0; i <= secret.length() -`
	`                    guess.length(); i++)`
	`{`
	`  if (secret.substring(i, i +`
	`          guess.length()).equals(guess))`
	`  …`

Some responses skipped overlapping occurrences of `guess` within `secret`.
`index = secret.indexOf(guess);`
`while (index != -1)`
`{`
`  index = secret.substring(index +`
`            guess.length()).indexOf(guess);`
`  …`
Some responses counted the same matching substring more than once.
`index = secret.indexOf(guess);`
`while (index != -1)`
`{`
`  index = secret.substring(index)`
`                            .indexOf(guess);`
`  …`
	`index = secret.indexOf(guess);`
	`while (index != -1)`
	`{`
	`  index = secret.substring(`
	`            index + 1).indexOf(guess);`
	`  …`

<table>
<tr><td>

Some responses modified persistent data (the value of `secret`) when iterating over substrings of `secret`.

```
index = secret.indexOf(guess);
while (index != -1)
{
  count++;
  secret = secret.substring(index);
  index = secret.indexOf(guess);
}
```

</td><td>

```
secretCopy = secret;
index = secretCopy.indexOf(guess);
while (index != -1)
{
  count++;
  secretCopy =
            secretCopy.substring(index);
  index = secretCopy.indexOf(guess);
}
```

</td></tr>
<tr><td>

Some responses failed to return the correct guess.

```
if (scoreGuess(guess1) >
                        scoreGuess(guess2))
{
  return guess2;
}
if (guess1.compareTo(guess2) > 0)
{
  return guess2;
}
```

</td><td>

```
if (scoreGuess(guess1) >
                        scoreGuess(guess2))
{
  return guess1;
}
if (guess1.compareTo(guess2) > 0)
{
  return guess1;
}
```

</td></tr>
<tr><td>

Some responses failed to cast a call to `Math.pow` as an `int` when calculating the square of `guess.length()`.

```
return numOcc *
               Math.pow(guess.length(), 2);
```

</td><td>

```
return numOcc * (int)
            Math.pow(guess.length(), 2);
```

</td></tr>
<tr><td>

Common Misconceptions/Knowledge Gaps

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

</td><td>

Responses that Demonstrate Understanding

</td></tr>
<tr><td>

Some responses failed to make an alphabetical comparison between the two guesses.

```
if (guess1.substring(i, i +
           guess2.length()).equals(guess2))
```

or

```
if (guess1.indexOf(guess2) > -1)
```

or

```
if (guess1 > guess2)
```

</td><td>

```
if (guess1.compareTo(guess2) > 0)
```

</td></tr>
<tr><td>

Some responses returned an `int` instead of a `String`.

```
int score1 = scoreGuess(guess1);
int score2 = scoreGuess(guess2);
if (score1 > score2)
{
  return score1;
}
```

</td><td>

```
if (scoreGuess(guess1) >
                        scoreGuess(guess2))
{
  return guess1;
}
```

</td></tr>
</table>

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 question.

- 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 methods 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:** Combined Tables

Max. Points: 9 **Mean Score:** 4.94

What were the responses to this question expected to demonstrate?

This question tested the student's ability to:

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

Students were asked to design the class `CombinedTable`, which represents a table composed of two single tables pushed together. The students were given a partial definition of the class `SingleTable`, which represents a table at a restaurant, to be used in their `CombinedTable` class design. Students were expected to demonstrate an understanding of class constructor and method header syntax. Additionally, students were expected to determine the data types and instance variables needed to track the information shown in the example. Students were then expected to correctly declare, initialize, access, and generate the appropriate values from their data members. Students were expected to properly protect the data members by declaring them as `private` and properly define the methods `canSeat` and `getDesirability`. Students had to recognize that they could not compute and store the desirability value in the constructor because the design of `SingleTable` allowed for a table view quality to change at any time; `getDesirability` always had to reflect the latest values of the `SingleTable` view quality.

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 two `SingleTable` instance variables, initialized them in the constructor, and used their accessor methods to compute whether a `CombinedTable` had enough seats for some group of people and to compute the desirability rating for that particular `CombinedTable`. It was possible to also compute the number of seats for the `CombinedTable` in the constructor because this value never changed.

While many responses used this strategy, they often made mistakes with the structural implementation details.

A great many responses saw students attempt to compute and store the desirability rating for the `CombinedTable`. Because a `SingleTable` object could have its view quality updated at any time, a stored desirability value will not reflect the updated state of the `CombinedTable`. It seems that these students did not understand the implications of the `setViewQuality` method.

There were a number of responses where students saw this as an inheritance problem. Many class headers extended `SingleTable`, often without penalty, as no other attempts to use an inheritance relationship occurred within the class design. There were some responses, however, that attempted to use the `super` keyword in the constructor and in the methods.

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*

Some responses extended the class when this was not an inheritance problem.
`public class CombinedTable extends SingleTable`
Some responses provided parameters for the class header.
`public class CombinedTable (SingleTable t1, SingleTable t2)`
Some responses omitted the word `class` in the class header.
`public CombinedTable`
Some responses omitted the class header completely.
	`public class CombinedTable`

Some responses provided a return type to the constructor.
`public void CombinedTable(SingleTable t1, SingleTable t2)`
Some responses omitted the parameter types.
`public CombinedTable(t1, t2)`
Some responses provided the wrong parameter types.
`public CombinedTable(Object t1, Object t2)`
Some responses omitted parameters.
`public CombinedTable()`
Some responses omitted the constructor.
	`public CombinedTable(SingleTable t1, SingleTable t2)`

Some responses declared `SingleTable` instance variables of the wrong type.
`private int table1;`
or
`private Table table1;`
or
`private Object table1;`
or
`private CombinedTable table1;`
Some responses omitted the keyword `private`.
`SingleTable table1;`
	`private SingleTable table1;`

Some responses declared local variables inside the constructor instead of instance variables.
`public CombinedTable(SingleTable t1,`
`                     SingleTable t2)`
`{`
`   SingleTable table1 = t1;`
`   SingleTable table2 = t2;`
`}`
Some responses failed to initialize any instance variables.
`public CombinedTable(SingleTable t1,`
`                     SingleTable t2)`
`{ }`
Some responses called `super()` inappropriately. Even if they had properly extended `SingleTable`, the constructors for `SingleTable` were undefined in the problem.
`private SingleTable table1;`
`private SingleTable table2;`
 	
`public CombinedTable(SingleTable t1,`
`                     SingleTable t2)`
`{`
`   super();`
`   table1 = t1;`
`   table2 = t2;`
`}`
or
`public CombinedTable(SingleTable t1,`
`                     SingleTable t2)`
`{`
`   super(t1, t2);`
`}`
	`private SingleTable table1;`
	`private SingleTable table2;`
	
	`public CombinedTable(SingleTable t1,`
	`                     SingleTable t2)`
	` {`
	`    table1 = t1;`
	`    table2 = t2;`
	` }`

Some responses reversed assignment of parameters to instance variables.
 	
`public CombinedTable(SingleTable t1,`
`                     SingleTable t2)`
`{`
`   t1 = table1;`
`   t2 = table2;`
`}`	

Some responses omitted the keyword `public`.
 	
`canSeat(int num)`
 	
Some responses omitted the return type, or used the wrong return type.
 	
`public canSeat(int num)`
`public int canSeat()`
 	
Some responses omitted the parameter.
 	
`public boolean canSeat()`
 	
Some responses specified the wrong parameter type or the wrong number of parameters.
 	
`public boolean canSeat(SingleTable t1)`
`public boolean canSeat(SingleTable t1,`
`                       SingleTable t2)`
	`public boolean canSeat(int num)`

Some responses failed to specify the return type or used the wrong return type.
 	
`public getDesirability()`
`public int getDesirability()`
 	
Some responses included one or more parameters.
 	
`public double getDesirability`
`                         (SingleTable t1)`
 	
`public double getDesirability`
`         (SingleTable t1, SingleTable t2)`
	`public double getDesirability()`

<table>
<tr><td><em>Common Misconceptions/Knowledge Gaps</em>

<em>Write program code to create objects of a class and call methods.</em></td><td><em>Responses that Demonstrate Understanding</em></td></tr>
<tr><td>Some responses passed a <code>SingleTable</code> as a parameter.
<pre>int seats = getNumSeats(table1) …</pre>
Some responses used <code>super</code> as the object reference.
<pre>int seats = super.getNumSeats() …</pre></td><td><pre>int seats = table1.getNumSeats() …</pre></td></tr>
<tr><td>Some responses called <code>SingleTable</code> methods with references that were not specifically <code>SingleTable</code> objects.
<pre>private Object table1;
private Table table2;

if (table1.getHeight() …
                      // Object reference
  desirability = table2.getViewQuality()…
                      // Table reference</pre></td><td><pre>private SingleTable table1;
private SingleTable table2;

if (table1.getHeight() …
  desirability = table2.getViewQuality()…</pre></td></tr>
<tr><td><em>Common Misconceptions/Knowledge Gaps</em>

<em>Write program code to satisfy methods using expressions and conditional statements.</em></td><td><em>Responses that Demonstrate Understanding</em></td></tr>
<tr><td>Some responses used incorrect logic in comparing the requested seats and the available seats.
<pre>public boolean canSeat(int toBeSeated)
{
  int available = t1.getNumSeats() +
                  t2.getNumSeats() - 2;

  if (toBeSeated > available)
  {
    return true;
  }
  return false;
}</pre>
or
<pre> if (toBeSeated < available)</pre>
or
<pre> if (toBeSeated == available)</pre>
Some responses failed to subtract 2 from the total seats of both tables.
<pre> int available = t1.getNumSeats() +
                  t2.getNumSeats();
  …</pre></td><td><pre>public boolean canSeat(int toBeSeated)
{
  int available = t1.getNumSeats() +
                  t2.getNumSeats() - 2;

  if (toBeSeated <= available)
  {
    return true;
  }
  return false;
}</pre></td></tr>
</table>

<table>
<tr>
<td>Some responses incorrectly calculated the average to be returned.
<pre>
double desirability =
   table1.getViewQuality() +
   table2.getViewQuality() / 2.0;
</pre>
Some responses had type confusion.
<pre>
int desirability =
   (table1.getViewQuality() …
</pre>
Some responses called only one of the two methods, usually computing desirability based on the height of the tables.
<pre>
double desirability =
   (table1.getHeight() + …
</pre>
</td>
<td>
<pre>
double desirability =
   (table1.getViewQuality() +
    table2.getViewQuality()) / 2.0;
</pre>
</td>
</tr>
<tr>
<td>Some responses returned the wrong value based on the comparison of table heights.
<pre>
double desirability =
   (table1.getViewQuality() +
    table2.getViewQuality()) / 2.0;

if (table1.getHeight() ==
    table2.getHeight())
{
   return desirability - 10;
}
else
{
   return desirability;
}
</pre>
Some responses returned some average, without a conditional test.
<pre>
double desirability =
   (table1.getViewQuality() +
    table2.getViewQuality()) / 2.0;

return desirability;
</pre>
</td>
<td>
<pre>
double desirability =
   (table1.getViewQuality() +
    table2.getViewQuality()) / 2.0;

if (table1.getHeight() ==
    table2.getHeight())
{
   return desirability;
}
else
{
   return desirability - 10;
}
</pre>
</td>
</tr>
</table>

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 5 would be helpful to scaffold students' understanding for the Class Design free-response question.

- The following AP Daily Videos and corresponding Topic Questions can be found in AP Classroom to support this Class Design free-response question:

    - *Write program code to define a new type by creating a class.* Topics in Unit 5.

    - *Write program code to create objects of a class and call methods.* Topics 2.3, 2.4, and 2.5.

    - *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:** Club Members

Max. Points: 9 **Mean Score:** 4.20

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

This question involved the manipulation of both a one-dimensional array containing `String` values and an `ArrayList` containing `MemberInfo` objects. Students were expected to write two methods in the enclosing `ClubMembers` class, making use of its `ArrayList` instance variable as well as two methods from the `MemberInfo` class.

In part (a) students were expected to write a loop to access each element of an array parameter. Inside the loop, students were expected to: (1) Construct a `MemberInfo` object using the `new` keyword and three parameters: a name from the array, `gradYear`, and `true`, in that order; (2) Add the constructed `MemberInfo` object to the `ClubMembers` instance variable `memberList`.

In part (b) students were asked to develop an algorithm to: (1) Identify club members who have graduated and are in good standing and add those club members to an `ArrayList` to be returned; (2) Remove from `memberList` those club members who have graduated, regardless of whether or not they are in good standing; and (3) Leave club members who have <u>not</u> yet graduated in `memberList`. Students had to create an `ArrayList` of `MemberInfo` objects to be returned and write a loop to access each element of the given `ArrayList` instance variable. Inside the loop, students had to call `getGradYear` and correctly compare the `int` return value to the `year` parameter. They also had to call `inGoodStanding` and use the `boolean` return value 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 create objects of a class and call methods.
Write program code to satisfy methods using expressions, conditional statements, and iterative statements.

Some responses failed to correctly identify graduating club members using the `<=` operator. Similarly, some responses failed to access `getGradYear` or `inGoodStanding` correctly, either by including parameter(s) in the method call or by calling the method on an object other than `MemberInfo`.

Many responses failed to nest or combine conditional statements to correctly distinguish three cases, based on graduation status and standing. However, most responses used conditional statements appropriately to distinguish two cases based on graduation status and/or standing.

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

While most responses could add an identified club member to an `ArrayList`, some did not create a new `ArrayList` first or did not initialize it correctly. Similarly, most responses iterated through the array and `ArrayList`, but some confused access to array elements with access to `ArrayList` elements. Many responses did not remove elements from the `ArrayList` correctly, either skipping elements using a forward traversal of the `ArrayList` and not accounting for the shift left of elements or trying to remove an element from the `ArrayList` being traversed using an enhanced `for` loop.

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 use the keyword `new` to construct a `MemberInfo` object.
 	
`memberList.add(MemberInfo(names[i],`
`gradYear, true);`
 	
or
 	
`MemberInfo temp = MemberInfo(names[i],`
`gradYear, true);`
	`memberList.add(new MemberInfo(names[i],`
	`gradYear, true));`
	
	or
	
	`MemberInfo temp = new MemberInfo(names[i],`
	`gradYear, true);`

Some responses failed to use the correct data type to construct a `MemberInfo` object.
 	
`Member temp = new Member(names[i],`
`gradYear, true);`
 	
or
 	
`ClubMembers temp = new`
`ClubMembers(names[i],`
`gradYear, true);`
	`MemberInfo temp = new MemberInfo(names[i],`
	`gradYear, true);`

Some responses failed to use the correct parameters to construct a `MemberInfo` object.
 	
`MemberInfo temp = new MemberInfo(names[i]);`
 	
or
 	
`MemberInfo temp = new MemberInfo(names[i],`
`gradYear);`
	`MemberInfo temp = new MemberInfo(names[i],`
	`gradYear, true);`

Some responses used the `equals` method to compare `boolean` values for equality.
 	
`if (memberList.get(i).`
`inGoodStanding().equals(true))`
	`if (memberList.get(i).inGoodStanding()`
	`== true)`
	
	or
	
	`if (memberList.get(i).inGoodStanding())`

<table>
<tr>
<td><em>Common Misconceptions/Knowledge Gaps</em>

<em>Write program code to satisfy methods using expressions, conditional statements, and iterative statements.</em></td>
<td><em>Responses that Demonstrate Understanding</em></td>
</tr>
<tr>
<td>Some responses failed to distinguish three cases based on graduation status and standing.
<pre>
ArrayList<MemberInfo> rem =
               new ArrayList<MemberInfo>();

for (int i = memberList.size() - 1; i >= 0;
                                    i--)
{
  if (memberList.get(i).getGradYear() <=
      year &&
      memberList.get(i).inGoodStanding())
  {
    rem.add(memberList.get(i));
  }
  else
  {
    memberList.remove(i);
  }
}
</pre>
Some responses failed to distinguish three cases by failing to access standing at all.
<pre>
ArrayList<MemberInfo> rem =
               new ArrayList<MemberInfo>();

for (int i = memberList.size() - 1; i >= 0;
                                    i--)
{
  if (memberList.get(i).getGradYear() <=
                                      year)
  {
    rem.add(memberList.get(i));
    memberList.remove(i);
  }
}
</pre></td>
<td><pre>
ArrayList<MemberInfo> rem =
               new ArrayList<MemberInfo>();

for (int i = memberList.size() - 1; i >= 0;
                                    i--)
{
  if (memberList.get(i).getGradYear() <=
      year &&
      memberList.get(i).inGoodStanding())
  {
    rem.add(memberList.get(i));
  }

  if (memberlist.get(i).getGradYear() <=
                                      year)
  {
    memberList.remove(i);
  }
}
</pre></td>
</tr>
<tr>
<td><em>Common Misconceptions/Knowledge Gaps</em>

<em>Write program code to create, traverse, and manipulate elements in 1D array or ArrayList objects.</em></td>
<td><em>Responses that Demonstrate Understanding</em></td>
</tr>
<tr>
<td>Some responses declared an <code>ArrayList</code> without initializing it.
<pre>
ArrayList<MemberInfo> removed;
</pre>
Some responses failed to declare an <code>ArrayList</code> at all.</td>
<td><pre>
ArrayList<MemberInfo> removed =
               new ArrayList<MemberInfo>();
</pre></td>
</tr>
</table>

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

```
for (int i = 0; i < names.length; i++)
{
  MemberInfo mem =
            new MemberInfo(names.get(i),
                           gradYear, true);
  …
```

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

```
for (int i = 0; i < names.length; i++)
{
  MemberInfo mem =
            new MemberInfo(names[0],
                           gradYear, true);
  …
```

Some responses accessed elements outside the bounds of the `names` array.

```
for (int i = 0; i <= names.length; i++)
{
  MemberInfo mem =
            new MemberInfo(names[i],
                           gradYear, true);
  …
```

```
for (int i = 0; i < names.length; i++)
{
  MemberInfo mem =
            new MemberInfo(names[i],
                           gradYear, true);
     …
```

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

```
for (int i = memberList.size() - 1; i >= 0;
                                    i--)
{
  if (memberList[i].getGradYear() <= year)
  …
```

```
for (int i = memberList.size() - 1; i >= 0;
                                    i--)
{
  if (memberList.get(i).getGradYear() <=
                                      year)
  …
```

Some responses accessed elements outside the bounds of the `ArrayList`.

```
for (int i = 0; i <= memberList.size();
                                       i++)
{
  if (memberList.get(i).getGradYear() <=
                                      year)
  …
```

```
for (int i = 0; i < memberList.size(); i++)
{
  if (memberList.get(i).getGradYear() <=
                                      year)
  …
```

Some responses accessed the entire `ArrayList` instead of a specific element.

```
for (int i = memberList.size() - 1; i >= 0;
                                    i--)
{
  if (memberList.getGradYear() <= year)
  …
```

```
for (int i = memberList.size() - 1; i >= 0;
                                    i--)
{
  if (memberList.get(i).getGradYear() <=
                                      year)
  …
```

Some responses removed elements inside an enhanced for loop, which throws an exception.

```
ArrayList<MemberInfo> rem =
               new ArrayList<MemberInfo>();

for (MemberInfo member : memberList)
{
  if (member.getGradYear() <= year)
  {
    if (member.inGoodStanding())
    {
      rem.add(member);
    }
    memberList.remove(i);
  }
}
```

```
ArrayList<MemberInfo> rem =
               new ArrayList<MemberInfo>();

for (int i = memberList.size() - 1; i >= 0;
                                    i--)
{
  if (memberList.get(i).getGradYear() <=
                                      year)
  {
    if (memberList.get(i).inGoodStanding())
    {
      rem.add(memberList.get(i));
    }
    memberList.remove(i);
  }
}
```

Some responses skipped elements using a forward traversal of the ArrayList and not accounting for the shift left of elements when trying to remove an element.

```
ArrayList<MemberInfo> rem =
               new ArrayList<MemberInfo>();

for (int i = 0; i < memberList.size(); i++)
{
  if (memberList.get(i).getGradYear() <=
                                      year)
  {
    if (memberList.get(i).inGoodStanding())
    {
      rem.add(memberList.get(i));
    }
    memberList.remove(i);
  }
}
```

```
ArrayList<MemberInfo> rem =
               new ArrayList<MemberInfo>();

for (int i = 0; i < memberList.size(); i++)
{
  if (memberList.get(i).getGradYear() <=
                                      year)
  {
    if (memberList.get(i).inGoodStanding())
    {
      rem.add(memberList.get(i));
    }
    memberList.remove(i);
    i--;
  }
}
```

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 6 and 7 would be helpful to scaffold students' understanding for the Array / `ArrayList` free-response question.

- 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 methods 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 or `ArrayList` objects.* Topics 6.1, 6.2, 7.2, 7.3, 7.4, and 7.5.

Question #4 **Task:** 2D Array **Topic:** Array Resizer

Max. Points: 9 **Mean Score:** 4.08

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

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

In part (a) students were asked to write a `boolean` method, `isNonZeroRow`, which returned `true` if and only if all elements in row `r` of a two-dimensional array, `array2D`, are not equal to zero. Students were expected to be able to use the parameters `r` and `array2D` to traverse the given row in the two-dimensional array and determine if there were any zeros in that row.

In part (b) students were asked to write a method called `resize`, which returned a new two-dimensional array containing only rows from `array2D` with all nonzero values. The elements in the new array should appear in the same order as the order in which they appeared in the original array. Students were expected to create a new two-dimensional array with the correct dimensions. Students were expected to use the method in part (a), `isNonZeroRow`, and the helper method, `numNonZeroRows`, provided in the class framework. Students were then expected to identify the rows that were nonzero rows and copy them to the new two-dimensional array.

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

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

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

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

While most responses were able to use a conditional statement to identify zeros in a row and call the method to identify these nonzero rows, some did not call the method properly. Some responses did not use the loop structure appropriate to iterate through the two-dimensional array.

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. Some chose to reimplement the helper method to get the correct number of rows. Many responses copied the rows from the given two-dimensional array by row reference instead of performing an element-by-element copy. Most responses were able to traverse all rows and columns of the array in part (b), but many had difficulty making sure they had a separate index to maintain the row in the new two-dimensional array when copying element-by-element. There were many responses that used `ArrayList` methods to access or copy to the two-dimensional array, including `add`, `remove`, and `get`.

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 called methods without parameters.
 	
`numNonZeroRows()`
`isNonZeroRow()`
 	
Some responses called methods with incorrect parameters.
 	
`isNonZeroRow(int[][] array2D)`
`isNonZeroRow(array2D[r])`
`isNonZeroRow(r)`
 	
Some responses called static methods on array objects.
 	
`array2D[r].isNonZeroRow()`
`array2D.numNonZeroRows()`
 	
Some responses called static methods on this object.
 	
`this.numNonZeroRows(array2D)`
`this.isNonZeroRow(array2D, r)`
	`numNonZeroRows(array2D)`
	`isNonZeroRow(array2D, r)`

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

Some responses incorrectly used an enhanced `for` loop to compare the item in the row with zero.
 	
`for (int i : array2D[r])`
`{`
`  if (array2D[r][i] == 0)`
`  …`
	`for (int i : array2D[r])`
	`{`
	`  if (i == 0)`
	`  …`

Some responses failed to initialize the index variable. `int index;` Some responses did not declare an index variable at all.	`int index = 0;`

<table>
<tr>
<th>

Common Misconceptions/Knowledge Gaps

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

</th>
<th>

Responses that Demonstrate Understanding

</th>
</tr>
<tr>
<td>

Some responses compared entire rows of a two-dimensional array instead of individual elements.

```
if (array2D[r] == 0)
```

</td>
<td>

```
if (array2D[r][c] == 0)
```

</td>
</tr>
<tr>
<td>

Some responses constructed the two-dimensional array with incorrect dimensions.

```
int[][] new2D = new int[][];
                       // missing dimensions
```

or

```
int[][] new2D = new int
  [numNonZeroRows(array2D)][array2D.length];
```

or

```
int[][] new2D = new int
        [array2D.length][array2D[0].length];
```

or

```
int[][] new2D = new int
               [numNonZeroRows(array2D)][3];
```

Some responses failed to use the keyword `new`.

```
int[][] new2D = int
   [numNonZeroRows(array2D)][array2D[0].length];
```

Some responses failed to construct a new array of type `int`.

```
int[][] new2D = new
   [numNonZeroRows(array2D)][array2D[0].length];
```

</td>
<td>

```
int[][] new2D = new int
   [numNonZeroRows(array2D)][array2D[0].length]
```

</td>
</tr>
<tr>
<td>

Some responses used incorrect bounds when accessing column elements of the two-dimensional array `array2D`.

```
int newRowIndex = 0;
for (int r = 0; r < array2D.length; r++)
{
  if (isNonZeroRow(array2D, r))
  {
    // wrong max columns below
    for (int c = 0; c < array2D.length;
                                       c++)
    {
      new2D[newRowIndex][c] =
                             array2D[r][c];
    }
    newRowIndex++;
  }
}
```

</td>
<td>

```
int newRowIndex = 0;
for (int r = 0; r < array2D.length; r++)
{
  if (isNonZeroRow(array2D, r))
  {
    for (int c = 0; c < array2D[r].length;
                                       c++)
    {
      new2D[newRowIndex][c] =
                             array2D[r][c];
    }
    newRowIndex++;
  }
}
```

</td>
</tr>
</table>

Some responses failed to maintain a distinct row index for the new 2D array.

```
for (int r = 0; r < array2D.length; r++)
{
  if (isNonZeroRow(array2D, r))
  {
    for (int c = 0;
               c < array2D[0].length; c++)
    {
      new2D[r][c] = array2D[r][c];
    }
  }
}
```

or

```
for (int r = 0; r < array2D.length; r++)
{
  if (isNonZeroRow(array2D, r))
  {
    new2D[r] = array2D[r];
  }
}
```

```
int newRowIndex = 0;
for (int r = 0; r < array2D.length; r++)
{
  if (isNonZeroRow(array2D, r))
  {
    for (int c = 0;
               c < array2D[r].length; c++)
    {
      new2D[newRowIndex][c] =
                            array2D[r][c];
    }
    newRowIndex++;
  }
}
```

or

```
int newRowIndex = 0;
for (int r = 0; r < array2D.length; r++)
{
  if (isNonZeroRow(array2D, r))
  {
    new2D[newRowIndex] = array2D[r];
    newRowIndex++;
  }
}
```

Some responses used `ArrayList` methods to manipulate the 2D array.

```
for (int r = 0; r < array2D.length; r++)
{
  if (isNonZeroRow(array2D, r))
  {
    new2D.add(array2D[r]);
  }
}
```

or

```
for (int r = 0; r < array2D.length; r++)
{
  if (isNonZeroRow(array2D, r))
  {
    new2D.add(r);
  }
}
```

or

```
for (int r = 0; r < array2D.length; r++)
{
  if (!isNonZeroRow(array2D, r))
  {
    array2D.remove(r);
  }
}
```

```
int newRowIndex = 0;
for (int r = 0; r < array2D.length; r++)
{
  if (isNonZeroRow(array2D, r))
  {
    for (int c = 0; c <
                array2D[r].length; c++)
    {
      new2D[newRowIndex][c] =
                           array2D[r][c];
    }
    newRowIndex++;
  }
}
```

or

```
int newRowIndex = 0;
for (int r = 0; r < array2D.length; r++)
{
  if (isNonZeroRow(array2D, r))
  {
    new2D[newRowIndex] = array2D[r];
    newRowIndex++;
  }
}
```

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 8 would be helpful to scaffold students' understanding for the 2D Array free-response question.

- 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 methods 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.