

## 2017 AP<sup>®</sup> COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

1. This question involves identifying and processing the digits of a non-negative integer. The declaration of the `Digits` class is shown below. You will write the constructor and one method for the `Digits` class.

```
public class Digits
{
    /** The list of digits from the number used to construct this object.
     *   The digits appear in the list in the same order in which they appear in the original number.
     */
    private ArrayList<Integer> digitList;

    /** Constructs a Digits object that represents num.
     *   Precondition: num >= 0
     */
    public Digits(int num)
    { /* to be implemented in part (a) */ }

    /** Returns true if the digits in this Digits object are in strictly increasing order;
     *   false otherwise.
     */
    public boolean isStrictlyIncreasing()
    { /* to be implemented in part (b) */ }
}
```

Part (a) begins on page 4.

## 2017 AP<sup>®</sup> COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the constructor for the `Digits` class. The constructor initializes and fills `digitList` with the digits from the non-negative integer `num`. The elements in `digitList` must be `Integer` objects representing single digits, and appear in the same order as the digits in `num`. Each of the following examples shows the declaration of a `Digits` object and the contents of `digitList` as initialized by the constructor.

### Example 1

```
Digits d1 = new Digits(15704);
```

d1:

	0	1	2	3	4
digitList:	1	5	7	0	4

### Example 2

```
Digits d2 = new Digits(0);
```

d2:

	0
digitList:	0

**WRITE YOUR SOLUTION ON THE NEXT PAGE.**

## 2017 AP<sup>®</sup> COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete the `Digits` constructor below.

```
/** Constructs a Digits object that represents num.  
 *   Precondition: num >= 0  
 */  
public Digits(int num)
```

Part (b) begins on page 6.

## 2017 AP<sup>®</sup> COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `Digits` method `isStrictlyIncreasing`. The method returns `true` if the elements of `digitList` appear in strictly increasing order; otherwise, it returns `false`. A list is considered strictly increasing if each element after the first is greater than (but not equal to) the preceding element.

The following table shows the results of several calls to `isStrictlyIncreasing`.

Method call	Value returned
<code>new Digits(7).isStrictlyIncreasing()</code>	<code>true</code>
<code>new Digits(1356).isStrictlyIncreasing()</code>	<code>true</code>
<code>new Digits(1336).isStrictlyIncreasing()</code>	<code>false</code>
<code>new Digits(1536).isStrictlyIncreasing()</code>	<code>false</code>
<code>new Digits(65310).isStrictlyIncreasing()</code>	<code>false</code>

**WRITE YOUR SOLUTION ON THE NEXT PAGE.**

## 2017 AP<sup>®</sup> COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `isStrictlyIncreasing` below.

```
/** Returns true if the digits in this Digits object are in strictly increasing order;  
 *      false otherwise.  
 */  
public boolean isStrictlyIncreasing()
```

## 2017 AP<sup>®</sup> COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

2. This question involves the design of a class that will be used to produce practice problems. The following `StudyPractice` interface represents practice problems that can be used to study some subject.

```
public interface StudyPractice
{
    /** Returns the current practice problem. */
    String getProblem();

    /** Changes to the next practice problem. */
    void nextProblem();
}
```

The `MultiPractice` class is a `StudyPractice` that produces multiplication practice problems. A `MultiPractice` object is constructed with two integer values: *first integer* and *initial second integer*. The first integer is a value that remains constant and is used as the first integer in every practice problem. The initial second integer is used as the starting value for the second integer in the practice problems. This second value is incremented for each additional practice problem that is produced by the class.

For example, a `MultiPractice` object created with the call `new MultiPractice(7, 3)` would be used to create the practice problems "7 TIMES 3", "7 TIMES 4", "7 TIMES 5", and so on.

In the `MultiPractice` class, the `getProblem` method returns a string in the format of "*first integer* TIMES *second integer*". The `nextProblem` method updates the state of the `MultiPractice` object to represent the next practice problem.

## 2017 AP<sup>®</sup> COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

The `Phrase` class includes the method `findNthOccurrence`, which returns the `nth` occurrence of a given string. You must use `findNthOccurrence` appropriately to receive full credit.

Complete method `replaceNthOccurrence` below.

```
/** Modifies the current phrase by replacing the nth occurrence of str with repl.
 * If the nth occurrence does not exist, the current phrase is unchanged.
 * Precondition: str.length() > 0 and n > 0
 */
public void replaceNthOccurrence(String str, int n, String repl)
```

Part (b) begins on page 14.

## 2017 AP<sup>®</sup> COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `Phrase` method `findLastOccurrence`. This method finds and returns the index of the last occurrence of a given string in `currentPhrase`. If the given string is not found, `-1` is returned. The following tables show several examples of the behavior of the method `findLastOccurrence`.

```
Phrase phrase1 = new Phrase("A cat ate late.");
```

Method call	Value returned
<code>phrase1.findLastOccurrence("at")</code>	11
<code>phrase1.findLastOccurrence("cat")</code>	2
<code>phrase1.findLastOccurrence("bat")</code>	-1

Class information for this question

```
public class Phrase
```

```
private String currentPhrase
public Phrase(String p)
public int findNthOccurrence(String str, int n)
public void replaceNthOccurrence(String str, int n, String repl)
public int findLastOccurrence(String str)
public String toString()
```

**WRITE YOUR SOLUTION ON THE NEXT PAGE.**



## 2017 AP<sup>®</sup> COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

You must use `findNthOccurrence` appropriately to receive full credit.

Complete method `findLastOccurrence` below.

```
/** Returns the index of the last occurrence of str in the current phrase;  
 * returns -1 if str is not found.  
 * Precondition: str.length() > 0  
 * Postcondition: the current phrase is not modified.  
 */  
public int findLastOccurrence(String str)
```

**2017 AP<sup>®</sup> COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS**

4. This question involves reasoning about a two-dimensional (2D) array of integers. You will write two static methods, both of which are in a single enclosing class named `Successors` (not shown). These methods process a 2D integer array that contains consecutive values. Each of these integers may be in any position in the 2D integer array. For example, the following 2D integer array with 3 rows and 4 columns contains the integers 5 through 16, inclusive.

2D Integer Array

	0	1	2	3
0	15	5	9	10
1	12	16	11	6
2	14	8	13	7

The following `Position` class is used to represent positions in the integer array. The notation  $(r, c)$  will be used to refer to a `Position` object with row  $r$  and column  $c$ .

```
public class Position
{
    /** Constructs a Position object with row r and column c. */
    public Position(int r, int c)
    {    /* implementation not shown */    }

    // There may be instance variables, constructors, and methods that are not shown.
}
```

- (a) Write a static method `findPosition` that takes an integer value and a 2D integer array and returns the position of the integer in the given 2D integer array. If the integer is not an element of the 2D integer array, the method returns `null`.

For example, assume that array `arr` is the 2D integer array shown at the beginning of the question.

- The call `findPosition(8, arr)` would return the `Position` object  $(2, 1)$  because the value 8 appears in `arr` at row 2 and column 1.
- The call `findPosition(17, arr)` would return `null` because the value 17 does not appear in `arr`.

# AP<sup>®</sup> COMPUTER SCIENCE A

## 2017 SCORING GUIDELINES

### Question 1: Digits

<b>Part (a)</b>	<code>Digits</code> constructor	<b>5 points</b>
-----------------	---------------------------------	-----------------

**Intent:** *Initialize instance variable using passed parameter*

- +1 Constructs `digitList`
- +1 Identifies a digit in `num`
- +1 Adds at least one identified digit to a list
- +1 Adds all identified digits to a list (*must be in context of a loop*)
- +1 **On exit:** `digitList` contains all and only digits of `num` in the correct order

<b>Part (b)</b>	<code>isStrictlyIncreasing</code>	<b>4 points</b>
-----------------	-----------------------------------	-----------------

**Intent:** *Determine whether or not elements in `digitList` are in increasing order*

- +1 Compares at least one identified consecutive pair of `digitList` elements
- +1 Determines if a consecutive pair of `digitList` is out of order (*must be in context of a `digitList` traversal*)
- +1 Compares all necessary consecutive pairs of elements (*no bounds errors*)
- +1 Returns `true` iff all consecutive pairs of elements are in order; returns `false` otherwise

<b>Question-Specific Penalties</b>
------------------------------------

- 2 (q) Uses confused identifier instead of `digitList`

# AP<sup>®</sup> COMPUTER SCIENCE A

## 2017 SCORING GUIDELINES

### Question 1: Scoring Notes

Part (a) Digits constructor			5 points
Points	Rubric Criteria	Responses earn the point if they ...	Responses will not earn the point if they ...
+1	Constructs digitList		<ul style="list-style-type: none"> <li>initialize a local variable instead of digitList</li> <li>create an ArrayList&lt;int&gt;</li> </ul>
+1	Identifies a digit in num	<ul style="list-style-type: none"> <li>identify one digit of num or a length one substring/character of the String representation of num</li> </ul>	<ul style="list-style-type: none"> <li>treat num itself as a String</li> <li>convert num to a String incorrectly</li> </ul>
+1	Adds at least one identified digit to a list	<ul style="list-style-type: none"> <li>call add for some ArrayList using the previously identified digit, even if that digit was identified incorrectly</li> </ul>	<ul style="list-style-type: none"> <li>add String or char to digitList without proper conversion to the correct type</li> </ul>
+1	Adds all identified digits to a list (must be in the context of a loop)	<ul style="list-style-type: none"> <li>call add for some ArrayList using previously identified digits, even if those digits were identified incorrectly</li> </ul>	<ul style="list-style-type: none"> <li>identify only 1 digit</li> </ul>
+1	On exit: digitList contains all and only digits of num in the correct order	<ul style="list-style-type: none"> <li>add to digitList even if it is not instantiated properly</li> </ul>	<ul style="list-style-type: none"> <li>obtain a list with the digits in reverse order</li> <li>omit one or more digits</li> <li>add extra digits</li> <li>mishandle edge case, e.g., 0 or 10</li> <li>make a bounds error processing the String representation of num</li> </ul>
Part (b) isStrictlyIncreasing			4 points
Points	Rubric Criteria	Responses earn the point if they ...	Responses will not earn the point if they ...
+1	Compares at least one identified consecutive pair of digitList elements	<ul style="list-style-type: none"> <li>compare two consecutive Integers using compareTo</li> <li>explicitly convert two consecutive Integers to ints and compare those with &gt;=, &lt;= etc.</li> <li>use auto-unboxing to convert two consecutive Integers to ints and compare those with &gt;=, &lt;= etc.</li> </ul>	<ul style="list-style-type: none"> <li>access digitList as an array or string</li> <li>fail to call .get()</li> <li>compare using !&gt;</li> </ul>
+1	Determines if a consecutive pair of digitList is out of order (must be in context of a digitList traversal)	<ul style="list-style-type: none"> <li>determine the correct relationship between the two compared consecutive elements, even if the syntax of the comparison is incorrect</li> </ul>	<ul style="list-style-type: none"> <li>fail to consider the case where the two elements are equal for the false case</li> </ul>
+1	Compares all necessary consecutive pairs of elements (no bounds errors)		<ul style="list-style-type: none"> <li>return early</li> </ul>
+1	Returns true iff all consecutive pairs of elements are in order; returns false otherwise	<ul style="list-style-type: none"> <li>compare consecutive pairs for inequality, but fail to consider the case when two elements are equal</li> </ul>	<ul style="list-style-type: none"> <li>return prematurely via if (...) return false; else return true;</li> </ul>

# AP<sup>®</sup> COMPUTER SCIENCE A

## 2017 SCORING GUIDELINES

### Question 1: Digits

Part (a)

```
public Digits(int num)
{
    digitList = new ArrayList<Integer>();

    if (num == 0)
    {
        digitList.add(new Integer(0));
    }

    while (num > 0)
    {
        digitList.add(0, new Integer(num % 10));
        num /= 10;
    }
}
```

Part (b)

```
public boolean isStrictlyIncreasing()
{
    for (int i = 0; i < digitList.size()-1; i++)
    {
        if (digitList.get(i).intValue() >= digitList.get(i+1).intValue())
        {
            return false;
        }
    }
    return true;
}
```

**Note:** The solutions shown above were written in compliance with the AP Java subset methods listed for `Integer` objects. Students were allowed to use the automatic "boxing" and "unboxing" of `Integer` objects in their solutions, which eliminates the need to use `"new Integer(...)"` in part (a) and `"intValue()"` in part (b).

These canonical solutions serve an expository role, depicting general approaches to solution. Each reflects only one instance from the infinite set of valid solutions. The solutions are presented in a coding style chosen to enhance readability and facilitate understanding.