

9 points

Question 2: Class

Canonical solution

```
public class Textbook extends Book
{
    private int edition;

    public Textbook(String tbTitle, double tbPrice,
                    int tbEdition)
    {
        super(tbTitle, tbPrice);
        edition = tbEdition;
    }

    public int getEdition()
    {
        return edition;
    }

    public boolean canSubstituteFor(Textbook other)
    {
        return other.getTitle().equals(getTitle()) &&
            edition >= other.getEdition();
    }

    public String getBookInfo()
    {
    }
}
```

9 points

Textbook

Scoring Criteria		Decision Rules	
1	Declares class header (must not be private): class Textbook extends Book		1 point
2	Declares constructor header: public Textbook(String ____, double ____, int ____)		1 point
3	Constructor calls <code>super</code> as the first line with the appropriate parameters		1 point
4	Declares appropriate <code>private</code> instance variable and uses appropriate parameter to initialize it	Responses will not earn the point if they - omit the keyword <code>private</code> - declare the variable outside the class, or in the class within a method or constructor - redeclare and use the instance variables of the superclass	1 point
5	Declares at least one required method and all declared headers are correct: public boolean canSubstituteFor(Textbook ____) public int getEdition() public String getBookInfo()	Responses will not earn the point if they exclude <code>public</code>	1 point
6	<code>getEdition</code> returns value of instance variable	Responses will not earn the point if they fail to create an instance variable for the edition	1 point
7	<code>canSubstituteFor</code> determines whether <code>true</code> or <code>false</code> should be returned based on comparison of book titles and editions (<i>algorithm</i>)	Responses can still earn the point even if they - fail to return (<i>return is not assessed for this method</i>) - access the edition without calling <code>getEdition</code> - redeclare and use the <code>title</code> variable of the superclass instead of calling <code>getTitle</code> Responses will not earn the point if they - fail to use <code>equals</code> - call <code>getTitle</code> incorrectly in either case	1 point
8	<code>getBookInfo</code> calls <code>super.getBookInfo</code>	Responses can still earn the point even if they - redeclare and use the instance variables of the superclass Responses will not earn the point if they - include parameters	1 point

9	Constructs information string	Responses can still earn the point even if they - call <code>super.getBookInfo</code> incorrectly - fail to call <code>super.getBookInfo</code> and access <code>title</code> and <code>price</code> directly - fail to return (<i>return is not assessed for this method</i>) Responses will not earn the point if they - omit the literal hyphen(s) in the constructed string - omit the edition in the constructed string - concatenate strings incorrectly	1 point
Question-specific penalties			
None			
Total for question 2			9 points

Applying the Scoring Criteria

Apply the question scoring criteria first, which always takes precedence. Penalty points can only be deducted in a part of the question that has earned credit via the question rubric. No part of a question (a, b, c) may have a negative point total. A given penalty can be assessed only once for a question, even if it occurs multiple times or in multiple parts of that question. A maximum of 3 penalty points may be assessed per question.

1-Point Penalty

- v) Array/collection access confusion (`[]` `get`)
- w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)
- x) Local variables used but none declared
- y) Destruction of persistent data (e.g., changing value referenced by parameter)
- z) Void method or constructor that returns a value

No Penalty

- Extraneous code with no side-effect (e.g., valid precondition check, no-op)
- Spelling/case discrepancies where there is no ambiguity*
- Local variable not declared provided other variables are declared in some part
- `private` or `public` qualifier on a local variable
- Missing `public` qualifier on class or constructor header
- Keyword used as an identifier
- Common mathematical symbols used for operators (`x • ÷ ≤ ≥ <> ≠`)
- `[]` vs. `()` vs. `<>`
- `=` instead of `==` and vice versa
- `length/size` confusion for array, `String`, `List`, or `ArrayList`; with or without `()`
- Extraneous `[]` when referencing entire array
- `[i,j]` instead of `[i][j]`
- Extraneous size in array declaration, e.g., `int[size] nums = new int[size];`
- Missing `;` where structure clearly conveys intent
- Missing `{ }` where indentation clearly conveys intent
- Missing `()` on parameter-less method or constructor invocations
- Missing `()` around `if` or `while` conditions

*Spelling and case discrepancies for identifiers fall under the “No Penalty” category only if the correction can be **unambiguously** inferred from context, for example, “`ArrayList`” instead of “`ArrayList`”. As a counterexample, note that if the code declares “`int G=99, g=0;`”, then uses “`while (G < 10)`” instead of “`while (g < 10)`”, the context does **not** allow for the reader to assume the use of the lower case variable.

AP[®] Computer Science A

Scoring Guidelines

2019 SCORING GUIDELINES

Apply the question assessment rubric first, which always takes precedence. Penalty points can only be deducted in a part of the question that has earned credit via the question rubric. No part of a question (a, b, c) may have a negative point total. A given penalty can be assessed only once for a question, even if it occurs multiple times or in multiple parts of that question. A maximum of 3 penalty points may be assessed per question.

1-Point Penalty

- v) Array/collection access confusion (`[] get`)
- w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)
- x) Local variables used but none declared
- y) Destruction of persistent data (e.g., changing value referenced by parameter)
- z) Void method or constructor that returns a value

No Penalty

- o Extraneous code with no side-effect (e.g., valid precondition check, no-op)
- o Spelling/case discrepancies where there is no ambiguity*
- o Local variable not declared provided other variables are declared in some part
- o `private` or `public` qualifier on a local variable
- o Missing `public` qualifier on class or constructor header
- o Keyword used as an identifier
- o Common mathematical symbols used for operators (`*` `*` `+` `≤` `≥` `<>` `≠`)
- o `[]` vs. `()` vs. `<>`
- o `=` instead of `==` and vice versa
- o `length/size` confusion for array, `String`, `List`, or `ArrayList`; with or without `()`
- o Extraneous `[]` when referencing entire array
- o `[i,j]` instead of `[i][j]`
- o Extraneous size in array declaration, e.g., `int[size] nums = new int[size];`
- o Missing `;` where structure clearly conveys intent
- o Missing `{ }` where indentation clearly conveys intent
- o Missing `()` on parameter-less method or constructor invocations
- o Missing `()` around `if` or `while` conditions

*Spelling and case discrepancies for identifiers fall under the “No Penalty” category only if the correction can be **unambiguously** inferred from context, for example, “`ArrayList`” instead of “`ArrayList`”. As a counterexample, note that if the code declares “`int G=99, g=0;`”, then uses “`while (G < 10)`” instead of “`while (g < 10)`”, the context does **not** allow for the reader to assume the use of the lower-case variable.

2019 SCORING GUIDELINES

Question 1: Calendar

Part (a)	<code>numberOfLeapYears</code>	5 points
-----------------	--------------------------------	-----------------

Intent: Return the number of leap years in a range

- +1 Initializes a numeric variable
- +1 Loops through each necessary year in the range
- +1 Calls `isLeapYear` on some valid year in the range
- +1 Updates count based on result of calling `isLeapYear`
- +1 Returns count of leap years

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

Intent: Return an integer representing the day of the week for a given date

- +1 Calls `firstDayOfYear`
- +1 Calls `dayOfYear`
- +1 Calculates the value representing the day of the week
- +1 Returns the calculated value

Question-Specific Penalties

- 1 (t) Static methods called with `this`.

2019 SCORING GUIDELINES

Question 1: Scoring Notes

Part (a) <code>numberOfLeapYears</code>		5 points	
Points	Rubric Criteria	Responses earn the point even if they...	Responses will not earn the point if they...
+1	Initializes a numeric variable		use the variable for loop control only
+1	Loops through each necessary year in the range		consider years outside the range
+1	Calls <code>isLeapYear</code> on some valid year in the range	do not use a loop	
+1	Updates count based on result of calling <code>isLeapYear</code>	- do not use a loop - do not initialize the counter	use result as a non-boolean
+1	Returns count of leap years	- loop from <code>year1</code> to <code>year2</code> incorrectly - do not initialize the counter	- do not use a loop - update or initialize the counter incorrectly - return early inside the loop
Part (b) <code>dayOfWeek</code>		4 points	
Points	Rubric Criteria	Responses earn the point even if they...	Responses will not earn the point if they...
+1	Calls <code>firstDayOfYear</code>		do not use the given year
+1	Calls <code>dayOfYear</code>		have arguments out of order
+1	Calculates the value representing the day of the week		make any error in the calculation
+1	Returns the calculated value	return the value from calling <code>firstDayOfYear</code> or <code>dayOfYear</code>	return a constant value

2019 SCORING GUIDELINES

Question 1: Calendar

Part (a)

```
public static int numberOfLeapYears(int year1, int year2)
{
    int count = 0;
    for (int y = year1; y <= year2; y++)
    {
        if (isLeapYear(y))
        {
            count++;
        }
    }
    return count;
}
```

Part (b)

```
public static int dayOfWeek(int month, int day, int year)
{
    int startDay = firstDayOfYear(year);
    int nthDay = dayOfYear(month, day, year);
    int returnDay = (startDay + nthDay - 1) % 7;
    return returnDay;
}
```

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.

2019 SCORING GUIDELINES

Question 2: Step Tracker

Class: StepTracker

9 points

Intent: Define implementation of a class to record fitness data

- +1** Declares all appropriate `private` instance variables
- +2** Constructor
 - +1** Declares header: `public StepTracker(int ____)`
 - +1** Uses parameter and appropriate values to initialize instance variables
- +3** `addDailySteps` method
 - +1** Declares header: `public void addDailySteps(int ____)`
 - +1** Identifies active days and increments count
 - +1** Updates other instance variables appropriately
- +1** `activeDays` method
 - +1** Declares and implements `public int activeDays()`
- +2** `averageSteps` method
 - +1** Declares header: `public double averageSteps()`
 - +1** Returns calculated `double` average number of steps

2019 SCORING GUIDELINES

Question 2: Scoring Notes

Class StepTracker		9 points	
Points	Rubric Criteria	Responses earn the point even if they...	Responses will not earn the point if they...
+1	Declares all appropriate private instance variables		- omit keyword <code>private</code> - declare variables outside the class
+2	Constructor		
+1	Declares header: <code>public StepTracker(int ____)</code>	omit keyword <code>public</code>	declare method <code>private</code>
+1	Uses parameter and appropriate values to initialize instance variables	initialize primitive instance variables to default values when declared	- fail to use the parameter to initialize some instance variable - fail to declare instance variables - initialize local variables instead of instance variables - assign variables to parameters
+3	addDailySteps method		
+1	Declares header: <code>public void addDailySteps(int ____)</code>	omit keyword <code>public</code>	declare method <code>private</code>
+1	Identifies active days and increments count	put valid comparison erroneously in some other method	- fail to use the parameter as part of the comparison - fail to increment a count of active days - fail to increment an instance variable - compare parameter to some numeric constant
+1	Updates other instance variables appropriately		- update another instance variable only on active days - update another instance variable inappropriately - fail to update appropriate instance variable - update a local variable
+1	activeDays method		
+1	Declares and implements <code>public int activeDays()</code>	return appropriate count of active days where the instance variables were updated improperly in <code>addDailySteps</code> or <code>activeDays</code>	- declare method <code>private</code> - return value that is not the number of active days - fail to return a value

2019 SCORING GUIDELINES

Question 2: Scoring Notes (continued)

Points	Rubric Criteria	Responses earn the point even if they...	Responses will not earn the point if they...
+2	averageSteps method		
+1	Declares header: <code>public double averageSteps()</code>	omit keyword <code>public</code>	declare method <code>private</code>
+1	Returns calculated <code>double average</code> number of steps	- maintain instance variables improperly but calculate appropriate average - fail to handle the special case where no days are tracked	- use integer division - calculate something other than steps divided by days - fail to return

2019 SCORING GUIDELINES

Question 2: Step Tracker

```

public class StepTracker
{
    private int minSteps;
    private int totalSteps;
    private int numDays;
    private int numActiveDays;

    public StepTracker(int threshold)
    {
        minSteps = threshold;
        totalSteps = 0;
        numDays = 0;
        numActiveDays = 0;
    }

    public void addDailySteps(int steps)
    {
        totalSteps += steps;
        numDays++;
        if (steps >= minSteps)
        {
            numActiveDays++;
        }
    }

    public int activeDays()
    {
        return numActiveDays;
    }

    public double averageSteps()
    {
        if (numDays == 0)
        {
            return 0.0;
        }
        else
        {
            return (double) totalSteps / numDays;
        }
    }
}

```

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.

2019 SCORING GUIDELINES

Question 3: Delimiters

Part (a)	getDelimitersList	4 points
-----------------	-------------------	-----------------

Intent: Store delimiters from an array in an `ArrayList`

- +1** Creates `ArrayList<String>`
- +1** Accesses all elements in array `tokens` (*no bounds errors*)
- +1** Compares strings in `tokens` with both instance variables (*must be in the context of a loop*)
- +1** Adds delimiters into `ArrayList` in original order

Part (b)	isBalanced	5 points
-----------------	------------	-----------------

Intent: Determine whether open and close delimiters in an `ArrayList` are balanced

- +1** Initializes accumulator(s)
- +1** Accesses all elements in `ArrayList delimiters` (*no bounds errors*)
- +1** Compares strings in `delimiters` with instance variables and updates accumulator(s) accordingly
- +1** Identifies and returns appropriate `boolean` value to implement one rule
- +1** Identifies and returns appropriate `boolean` values for all cases

2019 SCORING GUIDELINES

Question 3: Scoring Notes

Part (a) <code>getDelimitersList</code>		4 points	
Points	Rubric Criteria	Responses earn the point even if they...	Responses will not earn the point if they...
+1	Creates <code>ArrayList<String></code>	omit <code><String></code>	omit the keyword <code>new</code>
+1	Accesses all elements in array <code>tokens</code> (no bounds errors)	return incorrectly inside the loop	- treat <code>tokens</code> as a single string - access elements of <code>tokens</code> as if from an <code>ArrayList</code> (e.g., <code>tokens.get(i)</code>)
+1	Compares strings in <code>tokens</code> with both instance variables (must be in the context of a loop)	access elements of <code>tokens</code> as if from an <code>ArrayList</code> (e.g., <code>tokens.get(i)</code>)	- use <code>==</code> for string comparison - treat <code>tokens</code> as a single string
+1	Adds delimiters into <code>ArrayList</code> in original order	add a delimiter by accessing <code>tokens</code> incorrectly (e.g., <code>tokens.get(i)</code>)	- add a token that is not a delimiter - do not maintain the original delimiter order
Part (b) <code>isBalanced</code>		5 points	
Points	Rubric Criteria	Responses earn the point even if they...	Responses will not earn the point if they...
+1	Initializes accumulator(s)	initialize inside the loop	initialize an accumulator variable but don't update it
+1	Accesses all elements in <code>ArrayList</code> <code>delimiters</code> (no bounds errors)	return incorrectly inside the loop	access elements of <code>delimiters</code> as if from an array (e.g., <code>delimiters[i]</code>)
+1	Compares strings in <code>delimiters</code> with instance variables and updates accumulator(s) accordingly	access elements of <code>delimiters</code> as if from an array (e.g., <code>delimiters[i]</code>)	- use <code>==</code> for string comparison - adjust an accumulator without a guarding condition
+1	Identifies and returns appropriate <code>boolean</code> value to implement one rule	- check for more closing delimiters (inside a loop) and return <code>false</code> - return <code>true</code> if the number of open and close delimiters is the same, and <code>false</code> otherwise (after a loop)	
+1	Identifies and returns appropriate <code>boolean</code> values for all cases	have correct logic with the exception of a loop bounds error, accessing elements as if from an array, or using <code>==</code> for string comparison	- initialize accumulator inside a loop - fail to check for more closing delimiters inside a loop

2019 SCORING GUIDELINES

Question 3: Delimiters

Part (a)

```
public ArrayList<String> getDelimitersList(String[] tokens)
{
    ArrayList<String> d = new ArrayList<String>();
    for (String str : tokens)
    {
        if (str.equals(openDel) || str.equals(closeDel))
        {
            d.add(str);
        }
    }
    return d;
}
```

Part (b)

```
public boolean isBalanced(ArrayList<String> delimiters)
{
    int openCount = 0;
    int closeCount = 0;

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

        if (closeCount > openCount)
        {
            return false;
        }
    }

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

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.

2019 SCORING GUIDELINES

Question 4: Light Board

Part (a)	<code>LightBoard</code>	4 points
-----------------	-------------------------	-----------------

Intent: Define implementation of a constructor that initializes a 2D array of lights

- +1** Creates a new `boolean[numRows][numCols]` and assigns to instance variable `lights`
- +1** Accesses all elements in the created 2D array (*no bounds errors*)
- +1** Computes the 40% probability
- +1** Sets all values of 2D array based on computed probability

Part (b)	<code>evaluateLight</code>	5 points
-----------------	----------------------------	-----------------

Intent: Evaluate the status of a light in a 2D array of lights

- +1** Accesses an element of `lights` as a `boolean` value in an expression
- +1** Traverses specified `col` of a 2D array (*no bounds errors*)
- +1** Counts the number of `true` values in the traversal
- +1** Performs an even calculation and a multiple of three calculation
- +1** Returns `true` or `false` according to all three rules

Question-Specific Penalties

- 1** (z) Constructor returns a value
- 1** (y) Destruction of persistent data

2019 SCORING GUIDELINES

Question 4: Scoring Notes

Part (a) <code>LightBoard</code>		4 points	
Points	Rubric Criteria	Responses earn the point even if they...	Responses will not earn the point if they...
+1	Creates a new <code>boolean[numRows][numCols]</code> and assigns to instance variable <code>lights</code>		- initialize a local variable that is never assigned to <code>lights</code> - omit the keyword <code>new</code> - use a type other than <code>boolean</code>
+1	Accesses all elements in the created 2D array (<i>no bounds errors</i>)	fail to create <code>lights</code> but assume <code>lights[numRows][numCols]</code>	
+1	Computes the 40% probability	use <code>Math.random() <= .4</code>	incorrectly cast to <code>int</code>
+1	Sets all values of 2D array based on computed probability	only assign <code>true</code> values	- compute a single probability but use it multiple times - reverse the sense of the comparison when assigning
Part (b) <code>evaluateLight</code>		5 points	
Points	Rubric Criteria	Responses earn the point even if they...	Responses will not earn the point if they...
+1	Accesses an element of <code>lights</code> as a <code>boolean</code> value in an expression		access <code>lights</code> as a type other than <code>boolean</code>
+1	Traverses specified <code>col</code> of a 2D array (<i>no bounds errors</i>)		
+1	Counts the number of <code>true</code> values in the traversal	- access too many or too few items in a single column - access a single row instead of a single column	count an item more than once
+1	Performs an even calculation and a multiple of three calculation		use <code>/</code> instead of <code>%</code>
+1	Returns <code>true</code> or <code>false</code> according to all three rules	have an incorrect column count but use the correct logic	- fail to return a value in some case - implement counting loop more than once with one loop that is incorrect

2019 SCORING GUIDELINES**Question 4: Light Board***Part (a)*

```

public LightBoard(int numRows, int numCols)
{
    lights = new boolean[numRows][numCols];

    for (int r = 0; r < numRows; r++)
    {
        for (int c = 0; c < numCols; c++)
        {
            double rnd = Math.random();
            lights[r][c] = rnd < 0.4;
        }
    }
}

```

Part (b)

```

public boolean evaluateLight(int row, int col)
{
    int numOn = 0;

    for (int r = 0; r < lights.length; r++)
    {
        if (lights[r][col])
        {
            numOn++;
        }
    }

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

```

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.

2017

AP Computer Science A

Scoring Guidelines

2017 GENERAL SCORING GUIDELINES

Apply the question assessment rubric first, which always takes precedence. Penalty points can only be deducted in a part of the question that has earned credit via the question rubric. No part of a question (a, b, c) may have a negative point total. A given penalty can be assessed only once for a question, even if it occurs multiple times or in multiple parts of that question. A maximum of 3 penalty points may be assessed per question.

1-Point Penalty

- v) Array/collection access confusion (`[]` `get`)
- w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)
- x) Local variables used but none declared
- y) Destruction of persistent data (e.g., changing value referenced by parameter)
- z) Void method or constructor that returns a value

No Penalty

- o Extraneous code with no side-effect (e.g., valid precondition check, no-op)
- o Spelling/case discrepancies where there is no ambiguity*
- o Local variable not declared provided other variables are declared in some part
- o `private` or `public` qualifier on a local variable
- o Missing `public` qualifier on class or constructor header
- o Keyword used as an identifier
- o Common mathematical symbols used for operators (`*` `*` `+` `≤` `≥` `<>` `≠`)
- o `[]` vs. `()` vs. `<>`
- o `=` instead of `==` and vice versa
- o `length/size` confusion for array, `String`, `List`, or `ArrayList`; with or without `()`
- o Extraneous `[]` when referencing entire array
- o `[i,j]` instead of `[i][j]`
- o Extraneous size in array declaration, e.g., `int[size] nums = new int[size];`
- o Missing `;` where structure clearly conveys intent
- o Missing `{ }` where indentation clearly conveys intent
- o Missing `( )` on parameter-less method or constructor invocations
- o Missing `( )` around `if` or `while` conditions

*Spelling and case discrepancies for identifiers fall under the “No Penalty” category only if the correction can be **unambiguously** inferred from context, for example, “`ArayList`” instead of “`ArrayList`.” As a counterexample, note that if the code declares “`int G=99, g=0;`”, then uses “`while (G < 10)`” instead of “`while (g < 10)`”, the context does **not** allow for the reader to assume the use of the lower case variable.

2017 SCORING GUIDELINES

Question 1: Digits

Part (a)	Digits constructor	5 points
----------	--------------------	----------

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

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

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

Question-Specific Penalties

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

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		- initialize a local variable instead of digitList - create an ArrayList<int>
+1	Identifies a digit in num	identify one digit of num or a length one substring/character of the String representation of num	- treat num itself as a String - convert num to a String incorrectly
+1	Adds at least one identified digit to a list	call add for some ArrayList using the previously identified digit, even if that digit was identified incorrectly	add String or char to digitList without proper conversion to the correct type
+1	Adds all identified digits to a list (must be in the context of a loop)	call add for some ArrayList using previously identified digits, even if those digits were identified incorrectly	identify only 1 digit
+1	On exit: digitList contains all and only digits of num in the correct order	add to digitList even if it is not instantiated properly	- obtain a list with the digits in reverse order - omit one or more digits - add extra digits - mishandle edge case, e.g., 0 or 10 - make a bounds error processing the String representation of num
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	- compare two consecutive Integers using compareTo - explicitly convert two consecutive Integers to ints and compare those with >=, <= etc. - use auto-unboxing to convert two consecutive Integers to ints and compare those with >=, <= etc.	- access digitList as an array or string - fail to call .get() - compare using !>
+1	Determines if a consecutive pair of digitList is out of order (must be in context of a digitList traversal)	determine the correct relationship between the two compared consecutive elements, even if the syntax of the comparison is incorrect	fail to consider the case where the two elements are equal for the false case
+1	Compares all necessary consecutive pairs of elements (no bounds errors)		return early
+1	Returns true iff all consecutive pairs of elements are in order; returns false otherwise	compare consecutive pairs for inequality, but fail to consider the case when two elements are equal	return prematurely via if (...) return false; else return true;

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).

2017 SCORING GUIDELINES

Question 2: MultiPractice

Class:	MultiPractice	9 points
---------------	---------------	-----------------

Intent: Define implementation of class to produce multiplication practice problems

- +1** Declares header: `public class MultiPractice implements StudyPractice`
- +1** Declares all necessary `private` instance variables
- +2** Constructor
 - +1** Declares header: `public MultiPractice(int __, int __)`
 - +1** Initializes all instance variables using parameters
- +3** `getProblem` method
 - +1** Declares header: `public String getProblem()`
 - +1** Builds string with current values of instance variables
 - +1** Returns constructed string
- +2** `nextProblem` method
 - +1** Declares header: `public void nextProblem()`
 - +1** Updates instance variable(s) to reflect incremented second number

2017 SCORING GUIDELINES

Question 2: Scoring Notes

Class MultiPractice		9 points	
Points	Rubric Criteria	Responses earn the point if they ...	Responses will not earn the point if they ...
+1	Declares header: <code>public class MultiPractice implements StudyPractice</code>	omit keyword <code>public</code>	declare class <code>private</code>
+1	Declares all necessary <code>private</code> instance variables	declare the unchanging instance variable as <code>final</code>	- declare variables as <code>static</code> - omit keyword <code>private</code>
+2	Constructor		
+1	Declares header: <code>public MultiPractice (int __, int __)</code>	omit keyword <code>public</code>	
+1	Initializes all instance variables using parameters		- fail to declare nonlocal variables - initialize local variables instead of instance variables - assign variables to parameters
+3	<code>getProblem</code> method		
+1	Declares header: <code>public String getProblem()</code>		fail to declare method <code>public</code>
+1	Builds string with current values of instance variables	- write appropriate code in a method other than <code>getProblem</code> - make capitalization or spacing errors	- fail to declare nonlocal variables - fail to use instance variables - miscast <code>(String) intVar</code> - call <code>intVar.toString()</code>
+1	Returns constructed string		return a literal string
+2	<code>nextProblem</code> method		
+1	Declares header: <code>public void nextProblem()</code>		fail to declare method <code>public</code>
+1	Updates instance variable(s) to reflect incremented second number		fail to declare non-local variables

2017 SCORING GUIDELINES

Question 2: MultPractice

```
public class MultPractice implements StudyPractice
{
    private int first;
    private int second;

    public MultPractice(int num1, int num2)
    {
        first = num1;
        second = num2;
    }

    public String getProblem()
    {
        return first + " TIMES " + second;
    }

    public void nextProblem()
    {
        second++;
    }
}
```

2017 SCORING GUIDELINES

Question 3: PhraseEditor

Part (a)	<code>replaceNthOccurrence</code>	5 points
-----------------	-----------------------------------	-----------------

Intent: *Replace the n th occurrence of a given string with a given replacement*

- +1 Calls `findNthOccurrence` to find the index of the n th occurrence
- +1 Preserves `currentPhrase` only if n th occurrence does not exist
- +1 Identifies components of `currentPhrase` to retain (uses `substring` to extract before/after)
- +1 Creates replacement string using identified components and `repl`
- +1 Assigns replacement string to instance variable (`currentPhrase`)

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

Intent: *Return the index of the last occurrence of a given string*

- +1 Calls `findNthOccurrence` to find the index of the n th occurrence
- +1 Increments (or decrements) the value used as n when finding n th occurrence
- +1 Returns the index of the last occurrence, if it exists
- +1 Returns -1 only when no occurrences exist

Question-Specific Penalties

- 1 (q) Uses `currentPhrase.findNthOccurrence`
- 2 (r) Confused identifier instead of `currentPhrase`

2017 SCORING GUIDELINES

Question 3: Scoring Notes

Part (a) replaceNthOccurrence			5 points
Points	Rubric Criteria	Responses earn the point if they ...	Responses will not earn the point if they ...
+1	Calls findNthOccurrence to find the index of the nth occurrence	do not use the result of calling findNthOccurrence	
+1	Preserves currentPhrase only if nth occurrence does not exist		fail to use a conditional
+1	Identifies components of currentPhrase to retain (uses substring to extract before/after)	identify start and end of substring to be replaced	
+1	Creates replacement string using identified components and repl		create a replacement string that is out of order
+1	Assigns replacement string to instance variable (currentPhrase)		
Part (b) findLastOccurrence			4 points
Points	Rubric Criteria	Responses earn the point if they ...	Responses will not earn the point if they ...
+1	Calls findNthOccurrence to find the index of the nth occurrence	do not use the result of calling findNthOccurrence	- return currentPhrase.lastIndexOf(str); - call findNthOccurrence with an integer parameter of 0
+1	Increments (or decrements) the value used as n when finding nth occurrence	- return currentPhrase.lastIndexOf(str); - advance through currentPhrase searching for nth occurrence of str	
+1	Returns the index of the last occurrence, if it exists	- return currentPhrase.lastIndexOf(str); - compute the correct value to be returned in all cases, but no return statement exists for any case	- shorten string being searched - always return in first iteration of the loop
+1	Returns -1 only when no occurrences exist	return currentPhrase.lastIndexOf(str);	- compute the correct value to be returned in all cases, but no return statement exists for any case - always return in first iteration of the loop

2017 SCORING GUIDELINES

Question 3: PhraseEditor

Part (a)

```

public void replaceNthOccurrence(String str, int n, String repl)
{
    int loc = findNthOccurrence(str, n);

    if (loc != -1)
    {
        currentPhrase = currentPhrase.substring(0, loc) + repl +
                        currentPhrase.substring(loc + str.length());
    }
}

```

Part (b)

```

public int findLastOccurrence(String str)
{
    int n = 1;
    while (findNthOccurrence(str, n+1) != -1)
    {
        n++;
    }
    return findNthOccurrence(str, n);
}

```

2017 SCORING GUIDELINES

Question 4: Successor Array

Part (a)	findPosition	5 points
----------	--------------	----------

Intent: Find the position of a given integer in a 2D integer array

- +1 Accesses all necessary elements of `intArr` (no bounds errors)
- +1 Identifies `intArr` element equal to `num` (in context of an `intArr` traversal)
- +1 Constructs `Position` object with same row and column as identified `intArr` element
- +1 Selects constructed object when `intArr` element identified; `null` when not
- +1 Returns selected value

Part (b)	getSuccessorArray	4 points
----------	-------------------	----------

Intent: Create a successor array based on a 2D integer array

- +1 Creates 2D array of `Position` objects with same dimensions as `intArr`
- +1 Assigns a value to a location in 2D successor array using a valid call to `findPosition`
- +1 Determines the successor `Position` of an `intArr` element accessed by row and column (in context of `intArr` traversal)
- +1 Assigns all necessary locations in successor array with corresponding position object or `null` (no bounds errors)

Question-Specific Penalties

- 1 (s) Uses confused identifier `Arr`
- 1 (t) Uses `intArr[].length` as the number of columns
- 1 (u) Uses non-existent accessor methods from `Position`

2017 SCORING GUIDELINES

Question 4: Scoring Notes

Part (a) findPosition			5 points
Points	Rubric Criteria	Responses earn the point if they ...	Responses will not earn the point if they ...
+1	Accesses all necessary elements of <code>intArr</code> (no bounds errors)		- use <code>if (...) return;</code> else <code>return null;</code> inside loop - confuse row and column bounds - fail to traverse <code>intArr</code>
+1	Identifies <code>intArr</code> element equal to <code>num</code> (in context of an <code>intArr</code> traversal)		use <code>.equals</code> instead of <code>==</code>
+1	Constructs <code>Position</code> object with same row and column as identified <code>intArr</code> element		- omit keyword <code>new</code> - use <code>(r,c)</code> instead of <code>Position(r,c)</code>
+1	Selects constructed object when <code>intArr</code> element identified; <code>null</code> when not	- use "null" instead of <code>null</code> - construct a <code>String</code> object using row and column indices	- use <code>if (...) return;</code> else <code>return null;</code> inside loop - use <code>(r,c)</code> instead of <code>Position(r,c)</code>
+1	Returns selected value		
Part (b) getSuccessorArray			4 points
Points	Rubric Criteria	Responses earn the point if they ...	Responses will not earn the point if they ...
+1	Creates 2D array of <code>Position</code> objects with same dimensions as <code>intArr</code>		omit keyword <code>new</code>
+1	Assigns a value to a location in 2D successor array using a valid call to <code>findPosition</code>	call <code>Successors.findPosition(...)</code>	- reimplement the code from <code>findPosition</code> - call <code>findPosition</code> with a single argument - call <code>this.findPosition(...)</code>
+1	Determines the successor <code>Position</code> of an <code>intArr</code> element accessed by row and column (in context of <code>intArr</code> traversal)	reimplement the code from <code>findPosition</code>	- call <code>findPosition</code> using an integer that is not identified with a location in <code>intArr</code> - call <code>findPosition</code> with a single argument
+1	Assigns all necessary locations in successor array with corresponding position object or <code>null</code> (no bounds errors)	- use <code>SuccessorArray</code> dimensions correctly, even if <code>SuccessorArray</code> was not initialized properly - only assign non-null entries to <code>SuccessorArray</code>	- reimplement the code from <code>findPosition</code> but mishandle the null case. - fail to traverse <code>intArr</code>

Return is not assessed in Part (b).

2017 SCORING GUIDELINES**Question 4: Successor Array***Part (a)*

```

public static Position findPosition(int num, int[][] intArr)
{
    for (int row=0; row < intArr.length; row++)
    {
        for (int col=0; col < intArr[0].length; col++)
        {
            if (intArr[row][col] == num)
            {
                return new Position(row, col);
            }
        }
    }
    return null;
}

```

Part (b)

```

public static Position[][] getSuccessorArray(int[][] intArr)
{
    Position[][] newArr = new Position[intArr.length][intArr[0].length];

    for (int row=0; row < intArr.length; row++)
    {
        for (int col=0; col < intArr[0].length; col++)
        {
            newArr[row][col] = findPosition(intArr[row][col]+1, intArr);
        }
    }
    return newArr;
}

```

AP[®] Computer Science A

2016 Scoring Guidelines

© 2016 The College Board. College Board, Advanced Placement Program, AP, AP Central, and the acorn logo are registered trademarks of the College Board.

Visit the College Board on the Web: www.collegeboard.org.

AP Central is the official online home for the AP Program: apcentral.collegeboard.org.

2016 GENERAL SCORING GUIDELINES

Apply the question assessment rubric first, which always takes precedence. Penalty points can only be deducted in a part of the question that has earned credit via the question rubric. No part of a question (a, b, c) may have a negative point total. A given penalty can be assessed only once for a question, even if it occurs multiple times or in multiple parts of that question. A maximum of 3 penalty points may be assessed per question.

1-Point Penalty

- v) Array/collection access confusion (`[] get`)
- w) Extraneous code that causes side-effect (e.g., writing to output, failure to compile)
- x) Local variables used but none declared
- y) Destruction of persistent data (e.g., changing value referenced by parameter)
- z) Void method or constructor that returns a value

No Penalty

- o Extraneous code with no side-effect (e.g., precondition check, no-op)
- o Spelling/case discrepancies where there is no ambiguity*
- o Local variable not declared provided other variables are declared in some part
- o `private` or `public` qualifier on a local variable
- o Missing `public` qualifier on class or constructor header
- o Keyword used as an identifier
- o Common mathematical symbols used for operators (`*` `•` `÷` `≤` `≥` `<>` `≠`)
- o `[]` vs. `()` vs. `<>`
- o `=` instead of `==` and vice versa
- o `length/size` confusion for array, String, List, or ArrayList; with or without `()`
- o Extraneous `[]` when referencing entire array
- o `[i,j]` instead of `[i][j]`
- o Extraneous size in array declaration, e.g., `int[size] nums = new int[size];`
- o Missing `;` where structure clearly conveys intent
- o Missing `{ }` where indentation clearly conveys intent
- o Missing `()` on parameter-less method or constructor invocations
- o Missing `()` around `if` or `while` conditions

*Spelling and case discrepancies for identifiers fall under the “No Penalty” category only if the correction can be **unambiguously** inferred from context. For example, “ArrayList” instead of “Arraylist”. As a counter example, note that if the code declares “Bug bug;”, then uses “Bug.move()” instead of “bug.move()”, the context does **not** allow for the reader to assume the object instead of the class.

2016 SCORING GUIDELINES

Question 1: Random String Chooser

Part (a)	RandomStringChooser	7 points
----------	---------------------	----------

Intent: Define implementation of class to choose a random string

- +1** Uses correct class, constructor, and method headers
- +1** Declares appropriate `private` instance variable(s)
- +1** Initializes all instance variable(s) (*point lost if parameter not used in any initialization*)
- +4** Implements `getNext`
 - +1** Generates a random number in the proper range (*point lost for improper or missing cast*)
 - +1** Chooses a string from instance variable using generated random number
 - +1** Updates state appropriately (*point lost if constructor parameter is altered*)
 - +1** Returns chosen string or `"NONE"` as appropriate

Part (b)	RandomLetterChooser	2 points
----------	---------------------	----------

Intent: Define implementation of a constructor of a class that extends `RandomStringChooser`

- +1** `getSingleLetters(str)`
- +1** `super(getSingleLetters(str));` (*point lost if not first statement in constructor*)

2016 CANONICAL SOLUTIONS

Question 1: Random String Chooser

Part (a):

```
public class RandomStringChooser
{
    private List<String> words;

    public RandomStringChooser(String[] wordArray)
    {
        words = new ArrayList<String>();

        for (String singleWord : wordArray)
        {
            words.add(singleWord);
        }
    }

    public String getNext()
    {
        if (words.size() > 0)
        {
            return words.remove((int) (Math.random() * words.size()));
        }
        return "NONE";
    }
}
```

Part (b):

```
public RandomLetterChooser(String str)
{
    super(getSingleLetters(str));
}
```

2016 SCORING GUIDELINES

Question 2: Log Messages

Part (a) <code>LogMessage</code> constructor	2 points
---	-----------------

Intent: *Initialize instance variables using passed parameter*

- +1 Locates colon
- +1 Initializes instance variables with correct parts of the parameter

Part (b) <code>containsWord</code>	2 points
---	-----------------

Intent: *Determine whether description properly contains a keyword*

- +1 Identifies at least one properly-contained occurrence of keyword in description
- +1 Returns `true` if and only if `description` properly contains keyword
Returns `false` otherwise (*no bounds errors*)

Part (c) <code>removeMessages</code>	5 points
---	-----------------

Intent: *Remove log messages containing keyword from system log list and return these messages in a new list*

- +1 Accesses all items in `messageList` (*no bounds errors; point lost if no removal attempted*)
- +1 Identifies keyword-containing entry using `containsWord`
- +1 Adds all and only identified entries to new list (*point lost if original order not maintained*)
- +1 Removes all identified entries from `messageList` (*point lost if messageList reordered*)
- +1 Constructs and returns new `ArrayList<LogMessage>`

2016 CANONICAL SOLUTIONS

Question 2: Log Messages

Part (a):

```
public LogMessage(String message)
{
    int colon = message.indexOf(":");
    machineId = message.substring(0, colon);
    description = message.substring(colon + 1);
}
```

Part (b):

```
public boolean containsWord(String keyword)
{
    if (description.equals(keyword))
    {
        return true;
    }
    if (description.indexOf(keyword + " ") == 0)
    {
        return true;
    }
    if (description.indexOf(" " + keyword + " ") != -1)
    {
        return true;
    }
    if (description.length() > keyword.length())
    {
        if ((description.substring(description.length() -
                                   keyword.length() - 1).equals(
                                   " " + keyword)))
        {
            return true;
        }
    }
    return false;
}
```

Part (c):

```
public List<LogMessage> removeMessages(String keyword)
{
    List<LogMessage> removals = new ArrayList<LogMessage>();

    for (int i = 0; i < messageList.size(); i++)
    {
        if (messageList.get(i).containsWord(keyword))
        {
            removals.add(messageList.remove(i));
            i--;
        }
    }
    return removals;
}
```

2016 SCORING GUIDELINES

Question 3: Crossword

Part (a)	toBeLabeled	3 points
----------	-------------	----------

Intent: Return a *boolean* value indicating whether a crossword grid square should be labeled with a positive number

- +1 Checks `blackSquares[r][c]`
- +1 Checks for black square/border above and black square/border to the left (*no bounds errors*)
- +1 Returns `true` if square should be labeled with positive number; returns `false` otherwise

Part (b)	Crossword constructor	6 points
----------	-----------------------	----------

Intent: Initialize each square in a crossword puzzle grid to have a color (*boolean*) and an integer label

- +1 `puzzle = new Square[blackSquares.length][blackSquares[0].length];` (*or equivalent*)
- +1 Accesses all locations in `puzzle` (*no bounds errors*)
- +1 Calls `toBeLabeled` with appropriate parameters
- +1 Creates and assigns new `Square` to location in `puzzle`
- +1 Numbers identified squares consecutively, in row-major order, starting at 1
- +1 On exit: All squares in `puzzle` have correct color and number (*minor errors covered in previous points ok*)

Question-Specific Penalties

- 2 (p) Consistently uses incorrect name instead of `puzzle`
- 1 (q) Uses `array[].length` instead of `array[num].length`

2016 CANONICAL SOLUTIONS

Question 3: Crossword

Part (a):

```
private boolean toBeLabeled(int r, int c, boolean[][] blackSquares)
{
    return (!(blackSquares[r][c]) &&
            (r == 0 || c == 0 || blackSquares[r - 1][c] ||
             blackSquares[r][c - 1]));
}
```

Part (b):

```
public Crossword(boolean[][] blackSquares)
{
    puzzle = new Square[blackSquares.length][blackSquares[0].length];
    int num = 1;

    for (int r = 0; r < blackSquares.length; r++)
    {
        for (int c = 0; c < blackSquares[0].length; c++)
        {
            if (blackSquares[r][c])
            {
                puzzle[r][c] = new Square(true, 0);
            }
            else
            {
                if (toBeLabeled(r, c, blackSquares))
                {
                    puzzle[r][c] = new Square(false, num);
                    num++;
                }
                else
                {
                    puzzle[r][c] = new Square(false, 0);
                }
            }
        }
    }
}
```

2016 SCORING GUIDELINES

Question 4: String Formatter

Part (a)	<code>totalLetters</code>	2 points
-----------------	---------------------------	-----------------

Intent: Calculate the total number of letters in a list of words

- +1** Accesses all strings in `wordList` and adds length of each to accumulated count (*no bounds errors*)
- +1** Initializes and returns accumulated count

Part (b)	<code>basicGapWidth</code>	2 points
-----------------	----------------------------	-----------------

Intent: Calculate the minimum number of spaces (*basic gap width*) to be placed between each word in the formatted string

- +1** Calls `totalLetters` correctly and uses result
- +1** Returns correct calculated value

Part (c)	<code>format</code>	5 points
-----------------	---------------------	-----------------

Intent: Return a formatted string consisting of words from `wordList` separated by one or more spaces

- +1** Calls `basicGapWidth` and `leftoverSpaces` correctly and uses results
- +1** Adds all strings in `wordList` to formatted string in original order (*no bounds errors*)
- +1** Inserts *basicGapWidth* spaces between each pair of words in formatted string
- +1** Inserts one space between first *leftoverSpaces* pairs of words in formatted string
- +1** Initializes and returns formatted string (*no extra or deleted characters*)

2016 CANONICAL SOLUTIONS**Question 4: String Formatter**

Part (a):

```
public static int totalLetters(List<String> wordList)
{
    int total = 0;

    for (String word : wordList)
    {
        total += word.length();
    }
    return total;
}
```

Part (b):

```
public static int basicGapWidth(List<String> wordList, int formattedLen)
{
    return (formattedLen - totalLetters(wordList)) / (wordList.size()-1);
}
```

Part (c):

```
public static String format(List<String> wordList, int formattedLen)
{
    String formatted = "";
    int gapWidth = basicGapWidth(wordList, formattedLen);
    int leftovers = leftoverSpaces(wordList, formattedLen);

    for (int w = 0; w < wordList.size() - 1; w++)
    {
        formatted = formatted + wordList.get(w);
        for (int i = 0; i < gapWidth; i++)
        {
            formatted = formatted + " ";
        }
        if (leftovers > 0)
        {
            formatted = formatted + " ";
            leftovers--;
        }
    }
    formatted = formatted + wordList.get(wordList.size() - 1);

    return formatted;
}
```