

2024



AP[®] Computer Science A

Free-Response Questions



© 2024 College Board. College Board, Advanced Placement, AP, AP Central, and the acorn logo are registered trademarks of College Board. Visit College Board on the web: collegeboard.org.

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

COMPUTER SCIENCE A
SECTION II
Time—1 hour and 30 minutes
4 Questions

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA. You may plan your answers in this orange booklet, but no credit will be given for anything written in this booklet. **You will only earn credit for what you write in the separate Free Response booklet.**

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

GO ON TO THE NEXT PAGE.

1. This question simulates birds or possibly a bear eating at a bird feeder. The following `Feeder` class contains information about how much food is in the bird feeder and simulates how much food is eaten. You will write two methods of the `Feeder` class.

```
public class Feeder
{
    /**
     * The amount of food, in grams, currently in the bird feeder; initialized in the constructor and
     * always greater than or equal to zero
     */
    private int currentFood;

    /**
     * Simulates one day with numBirds birds or possibly a bear at the bird feeder,
     * as described in part (a)
     * Precondition: numBirds > 0
     */
    public void simulateOneDay(int numBirds)
    { /* to be implemented in part (a) */ }

    /**
     * Returns the number of days birds or a bear found food to eat at the feeder in this simulation,
     * as described in part (b)
     * Preconditions: numBirds > 0, numDays > 0
     */
    public int simulateManyDays(int numBirds, int numDays)
    { /* to be implemented in part (b) */ }

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

GO ON TO THE NEXT PAGE.

- (a) Write the `simulateOneDay` method, which simulates `numBirds` birds or possibly a bear at the feeder for one day. The method determines the amount of food taken from the feeder on this day and updates the `currentFood` instance variable. The simulation accounts for normal conditions, which occur 95% of the time, and abnormal conditions, which occur 5% of the time.

Under normal conditions, the simulation assumes that on any given day, only birds visit the feeder and that each bird at the feeder consumes the same amount of food. This standard amount consumed is between 10 and 50 grams of food, inclusive, in 1-gram increments. That is, on any given day, each bird might eat 10, 11, . . . , 49, or 50 grams of food. The amount of food eaten by each bird on a given day is randomly generated and each integer from 10 to 50, inclusive, has an equal chance of being chosen.

For example, a run of the simulation might predict that for a certain day under normal conditions, each bird coming to the feeder will eat 11 grams of food. If 10 birds come to the feeder on that day, then a total of 110 grams of food will be consumed.

If the simulated food consumed is greater than the amount of food in the feeder, the birds empty the feeder and the amount of food in the feeder at the end of the day is zero.

Under abnormal conditions, a bear empties the feeder and the amount of food in the feeder at the end of the day is zero.

The following examples show possible results of three calls to `simulateOneDay`.

- Example 1: If the feeder initially contains 500 grams of food, the call `simulateOneDay(12)` could result in 12 birds eating 20 grams of food each, leaving 260 grams of food in the feeder.
- Example 2: If the feeder initially contains 1,000 grams of food, the call `simulateOneDay(22)` could result in a bear eating all the food, leaving 0 grams of food in the feeder.
- Example 3: If the feeder initially contains 100 grams of food, the call `simulateOneDay(5)` could result in 5 birds attempting to eat 30 grams of food each. Since the feeder initially contains less than 150 grams of food, the feeder is emptied, leaving 0 grams of food in the feeder.

GO ON TO THE NEXT PAGE.

Complete the `simulateOneDay` method.

```
/**
 * Simulates one day with numBirds birds or possibly a bear at the bird feeder,
 * as described in part (a)
 * Precondition: numBirds > 0
 */
public void simulateOneDay(int numBirds)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class Feeder
private int currentFood
public void simulateOneDay(int numBirds)
public int simulateManyDays(int numBirds, int numDays)
```

GO ON TO THE NEXT PAGE.

- (b) Write the `simulateManyDays` method. The method uses `simulateOneDay` to simulate `numBirds` birds or a bear coming to the feeder on at most `numDays` consecutive days. The simulation returns the number of days that birds or a bear found food at the feeder.

Consider the following examples.

Value of <code>currentFood</code> and Method Call	Possible Outcomes and Resulting Return Value
<code>currentFood: 2400</code> <code>simulateManyDays(10, 4)</code>	Day 1: <code>simulateOneDay</code> leaves 2100 grams of food in the feeder. Day 2: <code>simulateOneDay</code> leaves 1650 grams of food in the feeder. Day 3: <code>simulateOneDay</code> leaves 1500 grams of food in the feeder. Day 4: <code>simulateOneDay</code> leaves 1260 grams of food in the feeder. The simulation returns 4 because, on all four days of the simulation, birds or a bear found food at the feeder. The instance variable <code>currentFood</code> has the value 1260.
<code>currentFood: 250</code> <code>simulateManyDays(10, 5)</code>	Day 1: <code>simulateOneDay</code> leaves 150 grams of food in the feeder. Day 2: <code>simulateOneDay</code> leaves 0 grams of food in the feeder. The simulation returns 2 because, on two of the five simulated days, birds or a bear found food at the feeder. The instance variable <code>currentFood</code> has the value 0.
<code>currentFood: 0</code> <code>simulateManyDays(5, 10)</code>	The simulation returns 0 because no food was found at the feeder on any day. The instance variable <code>currentFood</code> has the value 0.

GO ON TO THE NEXT PAGE.

Complete the `simulateManyDays` method. Assume that `simulateOneDay` works as intended, regardless of what you wrote in part (a). You must use `simulateOneDay` appropriately in order to receive full credit.

```
/**
 * Returns the number of days birds or a bear found food to eat at the feeder in this simulation,
 * as described in part (b)
 * Preconditions: numBirds > 0, numDays > 0
 */
public int simulateManyDays(int numBirds, int numDays)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class Feeder
private int currentFood
public void simulateOneDay(int numBirds)
public int simulateManyDays(int numBirds, int numDays)
```

GO ON TO THE NEXT PAGE.

2. This question involves a scoreboard for a game. The game is played between two teams who alternate turns so that at any given time, one team is active and the other team is inactive. During a turn, a team makes one or more plays. Each play can score one or more points and the team's turn continues, or the play can fail, in which case no points are scored and the team's turn ends. The `Scoreboard` class, which you will write, is used to keep track of the score in a game.

The `Scoreboard` class contains a constructor and two methods.

- The constructor has two parameters. The first parameter is a `String` containing the name of team 1, and the second parameter is a `String` containing the name of team 2. The game always begins with team 1 as the active team.
- The `recordPlay` method has a single nonnegative integer parameter that is equal to the number of points scored on a play or 0 if the play failed. If the play results in one or more points scored, the active team's score is updated and that team remains active. If the value of the parameter is 0, the active team's turn ends and the inactive team becomes the active team. The `recordPlay` method does not return a value.
- The `getScore` method has no parameters. The method returns a `String` containing information about the current state of the game. The returned string begins with the score of team 1, followed by a hyphen (" - "), followed by the score of team 2, followed by a hyphen, followed by the name of the team that is currently active.

GO ON TO THE NEXT PAGE.

The following table contains a sample code execution sequence and the corresponding results. The code execution sequence appears in a class other than `Scoreboard`.

Statement	Value Returned (blank if none)	Explanation
<code>String info;</code>		
<code>Scoreboard game = new Scoreboard("Red", "Blue");</code>		game is a new <code>Scoreboard</code> for a game played between team 1, whose name is "Red", and team 2, whose name is "Blue". The active team is set to team 1.
<code>info = game.getScore();</code>	"0-0-Red"	
<code>game.recordPlay(1);</code>		Team 1 earns 1 point because the game always begins with team 1 as the active team.
<code>info = game.getScore();</code>	"1-0-Red"	
<code>game.recordPlay(0);</code>		Team 1's play failed, so team 2 is now active.
<code>info = game.getScore();</code>	"1-0-Blue"	
<code>info = game.getScore();</code>	"1-0-Blue"	The score and state of the game are unchanged since the last call to <code>getScore</code> .
<code>game.recordPlay(3);</code>		Team 2 earns 3 points.
<code>info = game.getScore();</code>	"1-3-Blue"	
<code>game.recordPlay(1);</code>		Team 2 earns 1 point.
<code>game.recordPlay(0);</code>		Team 2's play failed, so team 1 is now active.
<code>info = game.getScore();</code>	"1-4-Red"	
<code>game.recordPlay(0);</code>		Team 1's play failed, so team 2 is now active.
<code>game.recordPlay(4);</code>		Team 2 earns 4 points.
<code>game.recordPlay(0);</code>		Team 2's play failed, so team 1 is now active.
<code>info = game.getScore();</code>	"1-8-Red"	
<code>Scoreboard match = new Scoreboard("Lions", "Tigers");</code>		match is a new and independent <code>Scoreboard</code> object.
<code>info = match.getScore();</code>	"0-0-Lions"	
<code>info = game.getScore();</code>	"1-8-Red"	

Write the complete `Scoreboard` class. Your implementation must meet all specifications and conform to the examples shown in the preceding table.

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

3. This question involves the manipulation and analysis of a list of words. The following `WordChecker` class contains an `ArrayList<String>` to be analyzed and methods that are used to perform the analysis. You will write two methods of the `WordChecker` class.

```
public class WordChecker
{
    /** Initialized in the constructor and contains no null elements */
    private ArrayList<String> wordList;

    /**
     * Returns true if each element of wordList (except the first) contains the previous
     * element as a substring and returns false otherwise, as described in part (a)
     * Precondition: wordList contains at least two elements.
     * Postcondition: wordList is unchanged.
     */
    public boolean isWordChain()
    { /* to be implemented in part (a) */ }

    /**
     * Returns an ArrayList<String> based on strings from wordList that start
     * with target, as described in part (b). Each element of the returned ArrayList has had
     * the initial occurrence of target removed.
     * Postconditions: wordList is unchanged.
     * Items appear in the returned list in the same order as they appear in wordList.
     */
    public ArrayList<String> createList(String target)
    { /* to be implemented in part (b) */ }

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

GO ON TO THE NEXT PAGE.

- (a) Write the `isWordChain` method, which determines whether each element of `wordList` (except the first) contains the previous element as a substring. The following table shows two sample `isWordChain` method calls.

<code>wordList</code>	<code>isWordChain</code> Return Value	Explanation
<code>["an", "band", "band", "abandon"]</code>	<code>true</code>	Each element contains the previous element as a substring.
<code>["to", "too", "stool", "tools"]</code>	<code>false</code>	"tools" does not contain the substring "stool".

Complete the `isWordChain` method.

```
/**
 * Returns true if each element of wordList (except the first) contains the previous
 * element as a substring and returns false otherwise, as described in part (a)
 * Precondition: wordList contains at least two elements.
 * Postcondition: wordList is unchanged.
 */
public boolean isWordChain()
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

- (b) Write the `createList` method, which creates and returns an `ArrayList<String>`. The method identifies strings in `wordList` that start with `target` and returns a new `ArrayList` containing each identified string without the starting occurrence of `target`. Elements must appear in the returned list in the same order as they appear in `wordList`.

Consider an example where `wordList` contains the following strings.

```
["catch", "bobcat", "catchacat", "cat", "at"]
```

The following table shows the `ArrayList` returned by some calls to `createList`. In all cases, `wordList` is unchanged.

Method Call	ArrayList Returned by <code>createList</code>	Explanation
<code>createList("cat")</code>	<code>["ch", "chacat", ""]</code>	Only "catch", "catchacat", and "cat" begin with "cat".
<code>createList("catch")</code>	<code>["", "acat"]</code>	Only "catch" and "catchacat" begin with "catch".
<code>createList("dog")</code>	<code>[]</code>	None of the words in <code>wordList</code> begin with "dog".

Complete the `createList` method.

```
/**
 * Returns an ArrayList<String> based on strings from wordList that start
 * with target, as described in part (b). Each element of the returned ArrayList has had
 * the initial occurrence of target removed.
 * Postconditions: wordList is unchanged.
 * Items appear in the returned list in the same order as they appear in wordList.
 */
public ArrayList<String> createList(String target)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class WordChecker
private ArrayList<String> wordList
public boolean isWordChain()
public ArrayList<String> createList(String target)
```

GO ON TO THE NEXT PAGE.

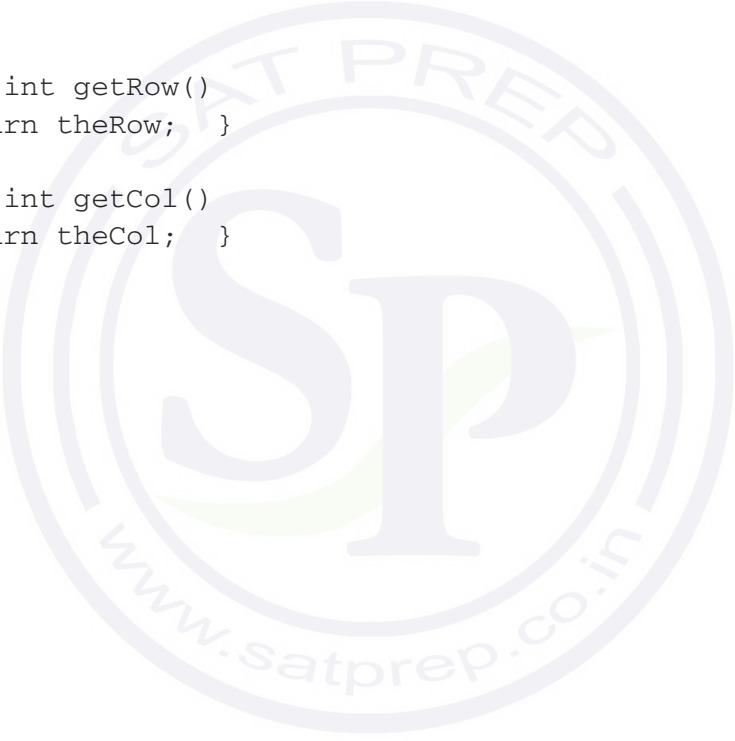
4. This question involves a path through a two-dimensional (2D) array of integers, where the path is based on the values of elements in the array. When an element of the 2D array is accessed, the first index is used to specify the row and the second index is used to specify the column. The following `Location` class represents a row and column position in the 2D array.

```
public class Location
{
    private int theRow;
    private int theCol;

    public Location(int r, int c)
    {
        theRow = r;
        theCol = c;
    }

    public int getRow()
    { return theRow; }

    public int getCol()
    { return theCol; }
}
```



GO ON TO THE NEXT PAGE.

The following `GridPath` class contains the 2D array and methods to use to determine a path through the array. You will write two methods of the `GridPath` class.

```
public class GridPath
{
    /** Initialized in the constructor with distinct values that never change */
    private int[][] grid;

    /**
     * Returns the Location representing a neighbor of the grid element at row and col,
     * as described in part (a)
     * Preconditions: row is a valid row index and col is a valid column index in grid.
     * row and col do not specify the element in the last row and last column of grid.
     */
    public Location getNextLoc(int row, int col)
    { /* to be implemented in part (a) */ }

    /**
     * Computes and returns the sum of all values on a path through grid, as described in
     * part (b)
     * Preconditions: row is a valid row index and col is a valid column index in grid.
     * row and col do not specify the element in the last row and last column of grid.
     */
    public int sumPath(int row, int col)
    { /* to be implemented in part (b) */ }

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

GO ON TO THE NEXT PAGE.

- (a) Write the `getNextLoc` method, which returns a `Location` object that represents the smaller of two neighbors of the grid element at `row` and `col`, according to the following rules.
- The two neighbors that are considered are the element below the given element and the element to the right of the given element, if they exist.
 - If both neighbors exist, the `Location` of the neighbor with the smaller value is returned. Two neighbors will always have different values.
 - If only one neighbor exists, the `Location` of the existing neighbor is returned.

For example, assume that `grid` contains the following values.

	0	1	2	3	4
0	12	3	4	13	5
1	11	21	2	14	16
2	7	8	9	15	0
3	10	17	20	19	1
4	18	22	30	25	6

The following table shows some sample calls to `getNextLoc`.

Method Call	Explanation
<code>getNextLoc(0, 0)</code>	Returns the neighbor to the right (the <code>Location</code> representing the element at row 0 and column 1), since <code>3 < 12</code>
<code>getNextLoc(1, 3)</code>	Returns the neighbor below (the <code>Location</code> representing the element at row 2 and column 3), since <code>15 < 14</code>
<code>getNextLoc(2, 4)</code>	Returns the neighbor below (the <code>Location</code> representing the element at row 3 and column 4), since the given element has no neighbor to the right
<code>getNextLoc(4, 3)</code>	Returns the neighbor to the right (the <code>Location</code> representing the element at row 4 and column 4), since the given element has no neighbor below

In the example, the `getNextLoc` method will never be called with row 4 and column 4, as those values would violate the precondition of the method.

GO ON TO THE NEXT PAGE.

Complete the getNextLoc method.

```
/**
 * Returns the Location representing a neighbor of the grid element at row and col,
 * as described in part (a)
 * Preconditions: row is a valid row index and col is a valid column index in grid.
 * row and col do not specify the element in the last row and last column of grid.
 */
public Location getNextLoc(int row, int col)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class Location
private int theRow
private int theCol
public Location(int r, int c)
public int getRow()
public int getCol()
public class GridPath
private int[][] grid
public Location getNextLoc(int row, int col)
public int sumPath(int row, int col)
```

GO ON TO THE NEXT PAGE.

- (b) Write the `sumPath` method, which returns the sum of all values on a path in `grid`. The path begins with the element at `row` and `col` and is determined by successive calls to `getNextLoc`. The path ends when the element in the last row and the last column of `grid` is reached.

For example, consider the following contents of `grid`. The shaded elements of `grid` represent the values on the path that results from the method call `sumPath(1, 1)`. The method call returns 19 because $3 + 2 + 9 + 4 + 0 + 1 = 19$.

	0	1	2	3	4
0	12	30	40	25	5
1	11	3	22	15	43
2	7	2	9	4	0
3	8	33	18	6	1

GO ON TO THE NEXT PAGE.

Write the `sumPath` method. Assume `getNextLoc` works as intended, regardless of what you wrote in part (a). You must use `getNextLoc` appropriately in order to receive full credit.

```
/**
 * Computes and returns the sum of all values on a path through grid, as described in
 * part (b)
 * Preconditions: row is a valid row index and col is a valid column index in grid.
 * row and col do not specify the element in the last row and last column of grid.
 */
public int sumPath(int row, int col)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class Location
private int theRow
private int theCol
public Location(int r, int c)
public int getRow()
public int getCol()
public class GridPath
private int[][] grid
public Location getNextLoc(int row, int col)
public int sumPath(int row, int col)
```

GO ON TO THE NEXT PAGE.



2023

AP[®]



AP[®] Computer Science A

Free-Response Questions



© 2023 College Board. College Board, Advanced Placement, AP, AP Central, and the acorn logo are registered trademarks of College Board. Visit College Board on the web: collegeboard.org.

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

COMPUTER SCIENCE A

SECTION II

Time—1 hour and 30 minutes

4 Questions

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA. You may plan your answers in this orange booklet, but no credit will be given for anything written in this booklet. **You will only earn credit for what you write in the separate Free Response booklet.**

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

GO ON TO THE NEXT PAGE.

1. This question involves the `AppointmentBook` class, which provides methods for students to schedule appointments with their teacher. Appointments can be scheduled during one of eight class periods during the school day, numbered 1 through 8. A requested appointment has a duration, which is the number of minutes the appointment will last. The 60 minutes within a period are numbered 0 through 59. In order for an appointment to be scheduled, the teacher must have a block of consecutive, available minutes that contains at least the requested number of minutes in a requested period. Scheduled appointments must start and end within the same period.



GO ON TO THE NEXT PAGE.

The AppointmentBook class contains two helper methods, isMinuteFree and reserveBlock. You will write two additional methods of the AppointmentBook class.

```
public class AppointmentBook
{
    /**
     * Returns true if minute in period is available for an appointment and returns
     * false otherwise
     * Preconditions: 1 <= period <= 8; 0 <= minute <= 59
     */
    private boolean isMinuteFree(int period, int minute)
    { /* implementation not shown */ }

    /**
     * Marks the block of minutes that starts at startMinute in period and
     * is duration minutes long as reserved for an appointment
     * Preconditions: 1 <= period <= 8; 0 <= startMinute <= 59;
     * 1 <= duration <= 60
     */
    private void reserveBlock(int period, int startMinute, int duration)
    { /* implementation not shown */ }

    /**
     * Searches for the first block of duration free minutes during period, as described in
     * part (a). Returns the first minute in the block if such a block is found or returns -1 if no
     * such block is found.
     * Preconditions: 1 <= period <= 8; 1 <= duration <= 60
     */
    public int findFreeBlock(int period, int duration)
    { /* to be implemented in part (a) */ }

    /**
     * Searches periods from startPeriod to endPeriod, inclusive, for a block
     * of duration free minutes, as described in part (b). If such a block is found,
     * calls reserveBlock to reserve the block of minutes and returns true; otherwise
     * returns false.
     * Preconditions: 1 <= startPeriod <= endPeriod <= 8; 1 <= duration <= 60
     */
    public boolean makeAppointment(int startPeriod, int endPeriod,
                                   int duration)
    { /* to be implemented in part (b) */ }

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

GO ON TO THE NEXT PAGE.

- (a) Write the `findFreeBlock` method, which searches `period` for the first block of free minutes that is `duration` minutes long. If such a block is found, `findFreeBlock` returns the first minute in the block. Otherwise, `findFreeBlock` returns `-1`. The `findFreeBlock` method uses the helper method `isMinuteFree`, which returns `true` if a particular minute is available to be included in a new appointment and returns `false` if the minute is unavailable.

Consider the following list of unavailable and available minutes in period 2.

Minutes in Period 2	Available?
0–9 (10 minutes)	No
10–14 (5 minutes)	Yes
15–29 (15 minutes)	No
30–44 (15 minutes)	Yes
45–49 (5 minutes)	No
50–59 (10 minutes)	Yes

The method call `findFreeBlock(2, 15)` would return 30 to indicate that a 15-minute block starting with minute 30 is available. No steps should be taken as a result of the call to `findFreeBlock` to mark those 15 minutes as unavailable.

The method call `findFreeBlock(2, 9)` would also return 30. Whenever there are multiple blocks that satisfy the requirement, the earliest starting minute is returned.

The method call `findFreeBlock(2, 20)` would return `-1`, since no 20-minute block of available minutes exists in period 2.

Complete method `findFreeBlock`. You must use `isMinuteFree` appropriately in order to receive full credit.

```
/**
 * Searches for the first block of duration free minutes during period, as described in
 * part (a). Returns the first minute in the block if such a block is found or returns -1 if no
 * such block is found.
 * Preconditions: 1 <= period <= 8; 1 <= duration <= 60
 */
public int findFreeBlock(int period, int duration)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

- (b) Write the `makeAppointment` method, which searches the periods from `startPeriod` to `endPeriod`, inclusive, for the earliest block of `duration` available minutes in the lowest-numbered period. If such a block is found, the `makeAppointment` method calls the helper method `reserveBlock` to mark the minutes in the block as unavailable and returns `true`. If no such block is found, the `makeAppointment` method returns `false`.

Consider the following list of unavailable and available minutes in periods 2, 3, and 4 and three successive calls to `makeAppointment`.

Period	Minutes	Available?
2	0–24 (25 minutes)	No
2	25–29 (5 minutes)	Yes
2	30–59 (30 minutes)	No
3	0–14 (15 minutes)	Yes
3	15–40 (26 minutes)	No
3	41–59 (19 minutes)	Yes
4	0–4 (5 minutes)	No
4	5–29 (25 minutes)	Yes
4	30–43 (14 minutes)	No
4	44–59 (16 minutes)	Yes

The method call `makeAppointment(2, 4, 22)` returns `true` and results in the minutes 5 through 26, inclusive, in period 4 being marked as unavailable.

The method call `makeAppointment(3, 4, 3)` returns `true` and results in the minutes 0 through 2, inclusive, in period 3 being marked as unavailable.

The method call `makeAppointment(2, 4, 30)` returns `false`, since there is no block of 30 available minutes in periods 2, 3, or 4.

The following shows the updated list of unavailable and available minutes in periods 2, 3, and 4 after the three example method calls are complete.

Period	Minutes	Available?
2	0–24 (25 minutes)	No
2	25–29 (5 minutes)	Yes
2	30–59 (30 minutes)	No
3	0–2 (3 minutes)	No
3	3–14 (12 minutes)	Yes
3	15–40 (26 minutes)	No
3	41–59 (19 minutes)	Yes
4	0–26 (27 minutes)	No
4	27–29 (3 minutes)	Yes
4	30–43 (14 minutes)	No
4	44–59 (16 minutes)	Yes

GO ON TO THE NEXT PAGE.

Complete method `makeAppointment`. Assume that `findFreeBlock` works as intended, regardless of what you wrote in part (a). You must use `findFreeBlock` and `reserveBlock` appropriately in order to receive full credit.

```
/**
 * Searches periods from startPeriod to endPeriod, inclusive, for a block
 * of duration free minutes, as described in part (b). If such a block is found,
 * calls reserveBlock to reserve the block of minutes and returns true; otherwise
 * returns false.
 * Preconditions: 1 <= startPeriod <= endPeriod <= 8; 1 <= duration <= 60
 */
public boolean makeAppointment(int startPeriod, int endPeriod,
                               int duration)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class AppointmentBook
{
    private boolean isMinuteFree(int period, int minute)
    private void reserveBlock(int period, int startMinute,
                              int duration)
    public int findFreeBlock(int period, int duration)
    public boolean makeAppointment(int startPeriod, int endPeriod,
                                   int duration)
```

GO ON TO THE NEXT PAGE.

2. This question involves methods that distribute text across lines of an electronic sign. The electronic sign and the text to be displayed on it are represented by the `Sign` class. You will write the complete `Sign` class, which contains a constructor and two methods.

The `Sign` class constructor has two parameters. The first parameter is a `String` that contains the message to be displayed on the sign. The second parameter is an `int` that contains the *width* of each line of the sign. The width is the positive maximum number of characters that can be displayed on a single line of the sign.

A sign contains as many lines as are necessary to display the entire message. The message is split among the lines of the sign without regard to spaces or punctuation. Only the last line of the sign may contain fewer characters than the width indicated by the constructor parameter.

The following are examples of a message displayed on signs of different widths. Assume that in each example, the sign is declared with the width specified in the first column of the table and with the message "Everything on sale, please come in", which contains 34 characters.

Width of the Sign	Sign Display
15	<div>Everything on s ale, please com e in</div>
17	<div>Everything on sal e, please come in</div>
40	<div>Everything on sale, please come in</div>

In addition to the constructor, the `Sign` class contains two methods.

The `numberOfLines` method returns an `int` representing the number of lines needed to display the text on the sign. In the previous examples, `numberOfLines` would return 3, 2, and 1, respectively, for the sign widths shown in the table.

The `getLines` method returns a `String` containing the message broken into lines separated by semicolons (;) or returns `null` if the message is the empty string. The constructor parameter that contains the message to be displayed will not include any semicolons. As an example, in the first row of the preceding table, `getLines` would return "Everything on s;ale, please com;e in". No semicolon should appear at the end of the `String` returned by `getLines`.

GO ON TO THE NEXT PAGE.

The following table contains a sample code execution sequence and the corresponding results. The code execution sequence appears in a class other than `Sign`.

Statement	Method Call Return Value (blank if none)	Explanation
<code>String str;</code>		
<code>int x;</code>		
<code>Sign sign1 = new Sign("ABC222DE", 3);</code>		The message for <code>sign1</code> contains 8 characters, and the sign has lines of width 3.
<code>x = sign1.numberOfLines();</code>	3	The sign needs three lines to display the 8-character message on a sign with lines of width 3.
<code>str = sign1.getLines();</code>	"ABC;222;DE"	Semicolons separate the text displayed on the first, second, and third lines of the sign.
<code>str = sign1.getLines();</code>	"ABC;222;DE"	Successive calls to <code>getLines</code> return the same value.
<code>Sign sign2 = new Sign("ABCD", 10);</code>		The message for <code>sign2</code> contains 4 characters, and the sign has lines of width 10.
<code>x = sign2.numberOfLines();</code>	1	The sign needs one line to display the 4-character message on a sign with lines of width 10.
<code>str = sign2.getLines();</code>	"ABCD"	No semicolon appears, since the text to be displayed fits on the first line of the sign.
<code>Sign sign3 = new Sign("ABCDEF", 6);</code>		The message for <code>sign3</code> contains 6 characters, and the sign has lines of width 6.
<code>x = sign3.numberOfLines();</code>	1	The sign needs one line to display the 6-character message on a sign with lines of width 6.
<code>str = sign3.getLines();</code>	"ABCDEF"	No semicolon appears, since the text to be displayed fits on the first line of the sign.
<code>Sign sign4 = new Sign("", 4);</code>		The message for <code>sign4</code> is an empty string.
<code>x = sign4.numberOfLines();</code>	0	There is no text to display.
<code>str = sign4.getLines();</code>	null	There is no text to display.
<code>Sign sign5 = new Sign("AB_CD_EF", 2);</code>		The message for <code>sign5</code> contains 8 characters, and the sign has lines of width 2.
<code>x = sign5.numberOfLines();</code>	4	The sign needs four lines to display the 8-character message on a sign with lines of width 2.
<code>str = sign5.getLines();</code>	"AB;_C;D_;EF"	Semicolons separate the text displayed on the four lines of the sign.

Write the complete `Sign` class. Your implementation must meet all specifications and conform to the examples shown in the preceding table.

GO ON TO THE NEXT PAGE.

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.



GO ON TO THE NEXT PAGE.

3. This question involves the analysis of weather data. The following `WeatherData` class has an instance variable, `temperatures`, which contains the daily high temperatures recorded on consecutive days at a particular location. The class also contains methods used to analyze that data. You will write two methods of the `WeatherData` class.

```
public class WeatherData
{
    /** Guaranteed not to be null and to contain only non-null entries */
    private ArrayList<Double> temperatures;

    /**
     * Cleans the data by removing from temperatures all values that are less than
     * lower and all values that are greater than upper, as described in part (a)
     */
    public void cleanData(double lower, double upper)
    { /* to be implemented in part (a) */ }

    /**
     * Returns the length of the longest heat wave found in temperatures, as described in
     * part (b)
     * Precondition: There is at least one heat wave in temperatures based on threshold.
     */
    public int longestHeatWave(double threshold)
    { /* to be implemented in part (b) */ }

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

GO ON TO THE NEXT PAGE.

- (a) Write the `cleanData` method, which modifies the `temperatures` instance variable by removing all values that are less than the `lower` parameter and all values that are greater than the `upper` parameter. The order of the remaining values in `temperatures` must be maintained.

For example, consider a `WeatherData` object for which `temperatures` contains the following.

99.1	142.0	85.0	85.1	84.6	94.3	124.9	98.0	101.0	102.5
------	-------	------	------	------	------	-------	------	-------	-------

The three shaded values shown would be removed by the method call `cleanData(85.0, 120.0)`.

99.1	142.0	85.0	85.1	84.6	94.3	124.9	98.0	101.0	102.5
------	-------	------	------	------	------	-------	------	-------	-------

The following shows the contents of `temperatures` after the three shaded values are removed as a result of the method call `cleanData(85.0, 120.0)`.

99.1	85.0	85.1	94.3	98.0	101.0	102.5
------	------	------	------	------	-------	-------

Complete method `cleanData`.

```
/**
 * Cleans the data by removing from temperatures all values that are less than
 * lower and all values that are greater than upper, as described in part (a)
 */
public void cleanData(double lower, double upper)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

- (b) Write the `longestHeatWave` method, which returns the length of the longest heat wave found in the `temperatures` instance variable. A heat wave is a sequence of two or more consecutive days with a daily high temperature greater than the parameter `threshold`. The `temperatures` instance variable is guaranteed to contain at least one heat wave based on the `threshold` parameter.

For example, consider the following contents of `temperatures`.

100.5	98.5	102.0	103.9	87.5	105.2	90.3	94.8	109.1	102.1	107.4	93.2
-------	------	-------	-------	------	-------	------	------	-------	-------	-------	------

In the following sample contents of `temperatures`, all heat waves based on the `threshold` temperature of 100.5 are shaded. The method call `longestHeatWave(100.5)` would return 3, which is the length of the longest heat wave.

100.5	98.5	102.0	103.9	87.5	105.2	90.3	94.8	109.1	102.1	107.4	93.2
-------	------	-------	-------	------	-------	------	------	-------	-------	-------	------

In the following sample contents of `temperatures`, all heat waves based on the `threshold` temperature of 95.2 are shaded. The method call `longestHeatWave(95.2)` would return 4, which is the length of the longest heat wave.

100.5	98.5	102.0	103.9	87.5	105.2	90.3	94.8	109.1	102.1	107.4	93.2
-------	------	-------	-------	------	-------	------	------	-------	-------	-------	------

Complete method `longestHeatWave`.

```
/**
 * Returns the length of the longest heat wave found in temperatures, as described in
 * part (b)
 * Precondition: There is at least one heat wave in temperatures based on threshold.
 */
public int longestHeatWave(double threshold)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class WeatherData
private ArrayList<Double> temperatures
public void cleanData(double lower, double upper)
public int longestHeatWave(double threshold)
```

GO ON TO THE NEXT PAGE.

4. This question involves pieces of candy in a box. The `Candy` class represents a single piece of candy.

```
public class Candy
{
    /** Returns a String representing the flavor of this piece of candy */
    public String getFlavor()
    { /* implementation not shown */ }

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

The `BoxOfCandy` class represents a candy box where the candy is arranged in a rectangular grid. The instance variable of the class, `box`, is a rectangular two-dimensional array of `Candy` objects. A location in the candy box may contain a piece of candy or may be empty. A piece of candy is represented by a `Candy` object. An empty location is represented by `null`.

You will write two methods of the `BoxOfCandy` class.

```
public class BoxOfCandy
{
    /** box contains at least one row and is initialized in the constructor. */
    private Candy[][] box;

    /**
     * Moves one piece of candy in column col, if necessary and possible, so that the box
     * element in row 0 of column col contains a piece of candy, as described in part (a).
     * Returns false if there is no piece of candy in column col and returns true otherwise.
     * Precondition: col is a valid column index in box.
     */
    public boolean moveCandyToFirstRow(int col)
    { /* to be implemented in part (a) */ }






    /**
     * Removes from box and returns a piece of candy with flavor specified by the parameter, or
     * returns null if no such piece is found, as described in part (b)
     */
    public Candy removeNextByFlavor(String flavor)
    { /* to be implemented in part (b) */ }

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

GO ON TO THE NEXT PAGE.

- (a) Write the `moveCandyToFirstRow` method, which attempts to ensure that the `box` element at row `0` and column `col` contains a piece of candy, using the following steps.
- If the element at row `0` and column `col` already contains a piece of candy, then `box` is unchanged and the method returns `true`.
 - If the element at row `0` and column `col` does not contain a piece of candy, then the method searches the remaining rows of column `col` for a piece of candy. If a piece of candy can be found in column `col`, it is moved to row `0`, its previous location is set to `null`, and the method returns `true`; otherwise, the method returns `false`.

In the following example, the grid represents the contents of `box`. An empty square in the grid is `null` in `box`. A non-empty square in the grid represents a `box` element that contains a `Candy` object. The string in the square of the grid indicates the flavor of the piece of candy.






	0	1	2
0		 "lime"	
1		 "orange"	
2			 "cherry"
3		 "lemon"	 "grape"

The method call `moveCandyToFirstRow(0)` returns `false` because the `box` element at row `0` and column `0` does not contain a piece of candy and there are no pieces of candy in column `0` that can be moved to row `0`. The contents of `box` are unchanged.






The method call `moveCandyToFirstRow(1)` returns `true` because the `box` element at row `0` and column `1` already contains a piece of candy. The contents of `box` are unchanged.

GO ON TO THE NEXT PAGE.

The method call `moveCandyToFirstRow(2)` moves one of the two pieces of candy in column 2 to row 0 of column 2, sets the previous location of the piece of candy that was moved to `null`, and returns `true`. The new contents of `box` could be either of the following.

	0	1	2
0		 "lime"	 "cherry"
1		 "orange"	
2			
3		 "lemon"	 "grape"

or

	0	1	2
0		 "lime"	 "grape"
1		 "orange"	
2			 "cherry"
3		 "lemon"	

Complete the `moveCandyToFirstRow` method.

```
/**
 * Moves one piece of candy in column col, if necessary and possible, so that the box
 * element in row 0 of column col contains a piece of candy, as described in part (a).
 * Returns false if there is no piece of candy in column col and returns true otherwise.
 * Precondition: col is a valid column index in box.
 */
public boolean moveCandyToFirstRow(int col)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class Candy
public String getFlavor()
public class BoxOfCandy
private Candy[][] box
public boolean moveCandyToFirstRow(int col)
public Candy removeNextByFlavor(String flavor)
```










GO ON TO THE NEXT PAGE.

- (b) Write the `removeNextByFlavor` method, which attempts to remove and return one piece of candy from the box. The piece of candy to be removed is the first piece of candy with a flavor equal to the parameter `flavor` that is encountered while traversing the candy box in the following order: the last row of the box is traversed from left to right, then the next-to-last row of the box is traversed from left to right, etc., until either a piece of candy with the desired flavor is found or until the entire candy box has been searched.









If the `removeNextByFlavor` method finds a `Candy` object with the desired flavor, the corresponding box element is assigned `null`, all other box elements are unchanged, and the removed `Candy` object is returned. Otherwise, `box` is unchanged and the method returns `null`.

The following examples show three consecutive calls to the `removeNextByFlavor` method. The traversal of the candy box always begins in the last row and first column of the box.

The following grid shows the contents of `box` before any of the `removeNextByFlavor` method calls.








	0	1	2	3	4
0	 "lime"	 "lime"		 "lemon"	
1	 "orange"			 "lime"	 "lime"
2	 "cherry"		 "lemon"		 "orange"

The method call `removeNextByFlavor("cherry")` removes and returns the `Candy` object located in row 2 and column 0. The following grid shows the updated contents of `box`.

	0	1	2	3	4
0	 "lime"	 "lime"		 "lemon"	
1	 "orange"			 "lime"	 "lime"
2			 "lemon"		 "orange"

GO ON TO THE NEXT PAGE.

The method call `removeNextByFlavor("lime")` removes and returns the `Candy` object located in row 1 and column 3. The following grid shows the updated contents of `box`.

	0	1	2	3	4
0	 "lime"	 "lime"		 "lemon"	
1	 "orange"				 "lime"
2			 "lemon"		 "orange"

The method call `removeNextByFlavor("grape")` returns `null` because no grape-flavored candy is found. The contents of `box` are unchanged.

Complete the `removeNextByFlavor` method.

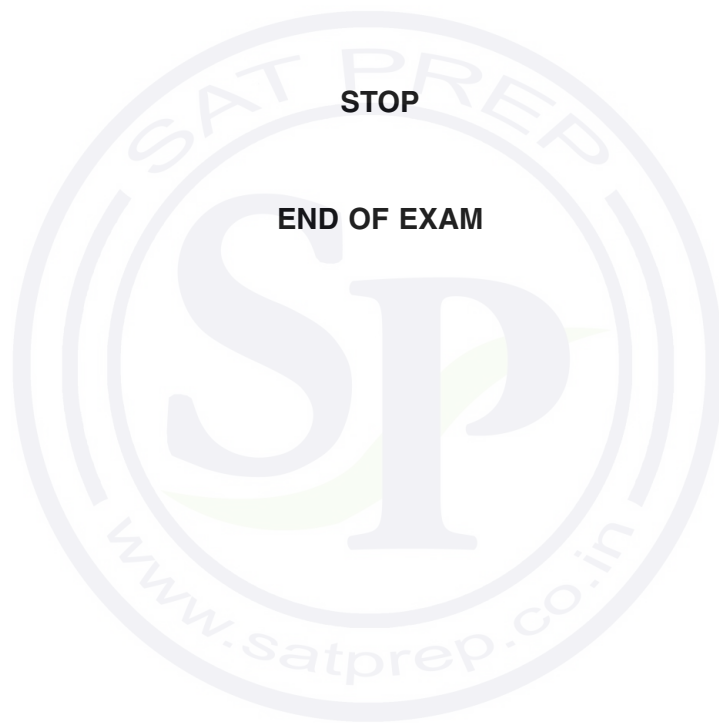
```
/**
 * Removes from box and returns a piece of candy with flavor specified by the parameter, or
 * returns null if no such piece is found, as described in part (b)
 */
public Candy removeNextByFlavor(String flavor)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class Candy
public String getFlavor()
public class BoxOfCandy
private Candy[][] box
public boolean moveCandyToFirstRow(int col)
public Candy removeNextByFlavor(String flavor)
```

GO ON TO THE NEXT PAGE.



2022

AP[®]

CollegeBoard

AP[®] Computer Science A

Free-Response Questions



COMPUTER SCIENCE A

SECTION II

Time—1 hour and 30 minutes

4 Questions

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA. You may plan your answers in this orange booklet, but no credit will be given for anything written in this booklet. **You will only earn credit for what you write in the separate Free Response booklet.**

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

GO ON TO THE NEXT PAGE.

1. This question involves simulation of the play and scoring of a single-player video game. In the game, a player attempts to complete three levels. A level in the game is represented by the `Level` class.

```
public class Level
{
    /** Returns true if the player reached the goal on this level and returns false otherwise */
    public boolean goalReached()
    { /* implementation not shown */ }

    /** Returns the number of points (a positive integer) recorded for this level */
    public int getPoints()
    { /* implementation not shown */ }

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



GO ON TO THE NEXT PAGE.

Play of the game is represented by the `Game` class. You will write two methods of the `Game` class.

```
public class Game
{
    private Level levelOne;
    private Level levelTwo;
    private Level levelThree;

    /** Postcondition: All instance variables have been initialized. */
    public Game()
    { /* implementation not shown */ }

    /** Returns true if this game is a bonus game and returns false otherwise */
    public boolean isBonus()
    { /* implementation not shown */ }

    /** Simulates the play of this Game (consisting of three levels) and updates all relevant
     * game data
     */
    public void play()
    { /* implementation not shown */ }

    /** Returns the score earned in the most recently played game, as described in part (a) */
    public int getScore()
    { /* to be implemented in part (a) */ }

    /** Simulates the play of num games and returns the highest score earned, as
     * described in part (b)
     * Precondition: num > 0
     */
    public int playManyTimes(int num)
    { /* to be implemented in part (b) */ }

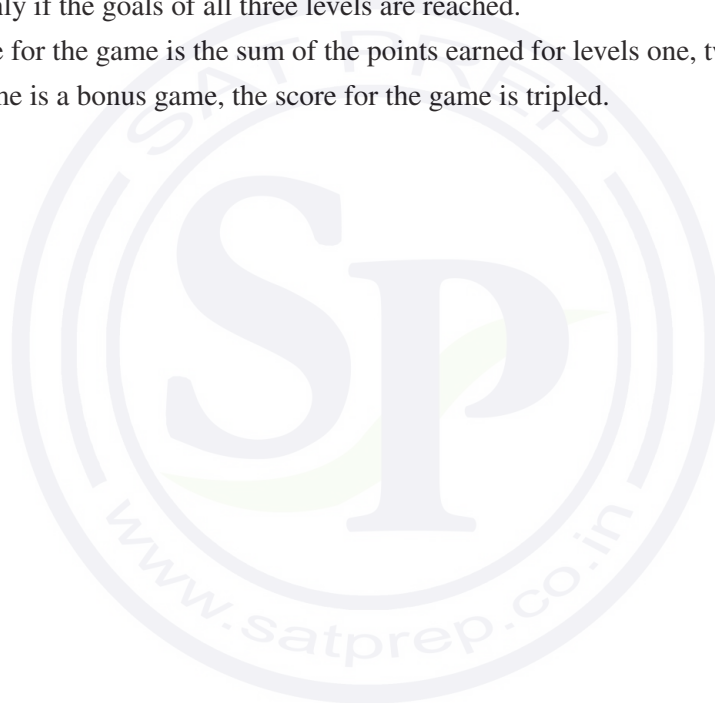
    // There may be instance variables, constructors, and methods that are not shown.
}
```

GO ON TO THE NEXT PAGE.

- (a) Write the `getScore` method, which returns the score for the most recently played game. Each game consists of three levels. The score for the game is computed using the following helper methods.
- The `isBonus` method of the `Game` class returns `true` if this is a bonus game and returns `false` otherwise.
 - The `goalReached` method of the `Level` class returns `true` if the goal has been reached on a particular level and returns `false` otherwise.
 - The `getPoints` method of the `Level` class returns the number of points recorded on a particular level. Whether or not recorded points are earned (included in the game score) depends on the rules of the game, which follow.

The score for the game is computed according to the following rules.

- Level one points are earned only if the level one goal is reached. Level two points are earned only if both the level one and level two goals are reached. Level three points are earned only if the goals of all three levels are reached.
- The score for the game is the sum of the points earned for levels one, two, and three.
- If the game is a bonus game, the score for the game is tripled.



GO ON TO THE NEXT PAGE.

The following table shows some examples of game score calculations.

	Level One Results	Level Two Results	Level Three Results	isBonus Return Value	Score Calculation
goalReached Return Value: getPoints Return Value:	true 200	true 100	true 500	true	$(200 + 100 + 500) \times 3 = 2,400$ The recorded points for levels one, two, and three are earned because the goals were reached in all three levels. The earned points are multiplied by 3 because isBonus returns true.
goalReached Return Value: getPoints Return Value:	true 200	true 100	false 500	false	$200 + 100 = 300$ The recorded points for level one and level two are earned because the goal was reached in levels one and two. The recorded points for level three are not earned because the goal was not reached in level three.
goalReached Return Value: getPoints Return Value:	true 200	false 100	true 500	true	$200 \times 3 = 600$ The recorded points for only level one are earned because the goal was not reached in level two. The earned points are multiplied by 3 because isBonus returns true.
goalReached Return Value: getPoints Return Value:	false 200	true 100	true 500	false	0 Because the goal in level one was not reached, no points are earned for any level.

Complete the `getScore` method.

```
/** Returns the score earned in the most recently played game, as described in part (a) */
public int getScore()
```

Begin your response at the top of a new page in the Free Response booklet and fill in the appropriate circle indicating the question number.
If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

- (b) Write the `playManyTimes` method, which simulates the play of `num` games and returns the highest game score earned. For example, if the four plays of the game that are simulated as a result of the method call `playManyTimes(4)` earn scores of 75, 50, 90, and 20, then the method should return 90.

Play of the game is simulated by calling the helper method `play`. Note that if `play` is called only one time followed by multiple consecutive calls to `getScore`, each call to `getScore` will return the score earned in the single simulated play of the game.

Complete the `playManyTimes` method. Assume that `getScore` works as intended, regardless of what you wrote in part (a). You must call `play` and `getScore` appropriately in order to receive full credit.

```
/** Simulates the play of num games and returns the highest score earned, as
 *   described in part (b)
 *   Precondition: num > 0
 */
public int playManyTimes(int num)
```

**Begin your response at the top of a new page in the Free Response booklet
and fill in the appropriate circle indicating the question number.
If there are multiple parts to this question, write the part letter with your response.**

Class information for this question

```
public class Level
public boolean goalReached()
public int getPoints()

public class Game
private Level levelOne
private Level levelTwo
private Level levelThree

public Game()
public boolean isBonus()
public void play()
public int getScore()
public int playManyTimes(int num)
```

GO ON TO THE NEXT PAGE.

2. The `Book` class is used to store information about a book. A partial `Book` class definition is shown.

```
public class Book
{
    /** The title of the book */
    private String title;

    /** The price of the book */
    private double price;

    /** Creates a new Book with given title and price */
    public Book(String bookTitle, double bookPrice)
    { /* implementation not shown */ }

    /** Returns the title of the book */
    public String getTitle()
    { return title; }

    /** Returns a string containing the title and price of the Book */
    public String getBookInfo()
    {
        return title + "-" + price;
    }

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

You will write a class `Textbook`, which is a subclass of `Book`.

A `Textbook` has an edition number, which is a positive integer used to identify different versions of the book. The `getBookInfo` method, when called on a `Textbook`, returns a string that also includes the edition information, as shown in the example.

Information about the book title and price must be maintained in the `Book` class. Information about the edition must be maintained in the `Textbook` class.

The `Textbook` class contains an additional method, `canSubstituteFor`, which returns `true` if a `Textbook` is a valid substitute for another `Textbook` and returns `false` otherwise. The current `Textbook` is a valid substitute for the `Textbook` referenced by the parameter of the `canSubstituteFor` method if the two `Textbook` objects have the same title and if the edition of the current `Textbook` is greater than or equal to the edition of the parameter.

GO ON TO THE NEXT PAGE.

The following table contains a sample code execution sequence and the corresponding results. The code execution sequence appears in a class other than `Book` or `Textbook`.

Statement	Value Returned (blank if no value)	Class Specification
<code>Textbook bio2015 = new Textbook("Biology", 49.75, 2);</code>		bio2015 is a <code>Textbook</code> with a title of "Biology", a price of 49.75, and an edition of 2.
<code>Textbook bio2019 = new Textbook("Biology", 39.75, 3);</code>		bio2019 is a <code>Textbook</code> with a title of "Biology", a price of 39.75, and an edition of 3.
<code>bio2019.getEdition();</code>	3	The edition is returned.
<code>bio2019.getBookInfo();</code>	"Biology-39.75-3"	The formatted string containing the title, price, and edition of bio2019 is returned.
<code>bio2019. canSubstituteFor(bio2015);</code>	true	bio2019 is a valid substitute for bio2015, since their titles are the same and the edition of bio2019 is greater than or equal to the edition of bio2015.
<code>bio2015. canSubstituteFor(bio2019);</code>	false	bio2015 is not a valid substitute for bio2019, since the edition of bio2015 is less than the edition of bio2019.
<code>Textbook math = new Textbook("Calculus", 45.25, 1);</code>		math is a <code>Textbook</code> with a title of "Calculus", a price of 45.25, and an edition of 1.
<code>bio2015. canSubstituteFor(math);</code>	false	bio2015 is not a valid substitute for math, since the title of bio2015 is not the same as the title of math.

Write the complete `Textbook` class. Your implementation must meet all specifications and conform to the examples shown in the preceding table.

Begin your response at the top of a new page in the Free Response booklet and fill in the appropriate circle indicating the question number.
If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

3. Users of a website are asked to provide a review of the website at the end of each visit. Each review, represented by an object of the `Review` class, consists of an integer indicating the user's rating of the website and an optional `String` comment field. The comment field in a `Review` object ends with a period ("."), exclamation point ("!"), or letter, or is a `String` of length 0 if the user did not enter a comment.

```
public class Review
{
    private int rating;
    private String comment;

    /** Precondition: r >= 0
     *      c is not null.
     */
    public Review(int r, String c)
    {
        rating = r;
        comment = c;
    }

    public int getRating()
    {
        return rating;
    }

    public String getComment()
    {
        return comment;
    }

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

GO ON TO THE NEXT PAGE.

The `ReviewAnalysis` class contains methods used to analyze the reviews provided by users. You will write two methods of the `ReviewAnalysis` class.

```
public class ReviewAnalysis
{
    /** All user reviews to be included in this analysis */
    private Review[] allReviews;

    /** Initializes allReviews to contain all the Review objects to be analyzed */
    public ReviewAnalysis()
    { /* implementation not shown */ }

    /** Returns a double representing the average rating of all the Review objects to be
     * analyzed, as described in part (a)
     * Precondition: allReviews contains at least one Review.
     * No element of allReviews is null.
     */
    public double getAverageRating()
    { /* to be implemented in part (a) */ }

    /** Returns an ArrayList of String objects containing formatted versions of
     * selected user comments, as described in part (b)
     * Precondition: allReviews contains at least one Review.
     * No element of allReviews is null.
     * Postcondition: allReviews is unchanged.
     */
    public ArrayList<String> collectComments()
    { /* to be implemented in part (b) */ }
}
```

GO ON TO THE NEXT PAGE.

- (a) Write the `ReviewAnalysis` method `getAverageRating`, which returns the average rating (arithmetic mean) of all elements of `allReviews`. For example, `getAverageRating` would return 3.4 if `allReviews` contained the following `Review` objects.

0	1	2	3	4
4 "Good! Thx"	3 "OK site"	5 "Great!"	2 "Poor! Bad."	3 ""

Complete method `getAverageRating`.

```
/** Returns a double representing the average rating of all the Review objects to be
 * analyzed, as described in part (a)
 * Precondition: allReviews contains at least one Review.
 * No element of allReviews is null.
 */
public double getAverageRating()
```

Begin your response at the top of a new page in the Free Response booklet and fill in the appropriate circle indicating the question number.
If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class Review
private int rating
private String comment
public Review(int r, String c)
public int getRating()
public String getComment()
public class ReviewAnalysis
private Review[] allReviews
public ReviewAnalysis()
public double getAverageRating()
public ArrayList<String> collectComments()
```

GO ON TO THE NEXT PAGE.

- (b) Write the `ReviewAnalysis` method `collectComments`, which collects and formats only comments that contain an exclamation point. The method returns an `ArrayList` of `String` objects containing copies of user comments from `allReviews` that contain an exclamation point, formatted as follows. An empty `ArrayList` is returned if no comment in `allReviews` contains an exclamation point.
- The `String` inserted into the `ArrayList` to be returned begins with the index of the `Review` in `allReviews`.
 - The index is immediately followed by a hyphen (" - ").
 - The hyphen is followed by a copy of the original comment.
 - The `String` must end with either a period or an exclamation point. If the original comment from `allReviews` does not end in either a period or an exclamation point, a period is added.

The following example of `allReviews` is repeated from part (a).

0	1	2	3	4
4 "Good! Thx"	3 "OK site"	5 "Great!"	2 "Poor! Bad."	3 ""

The following `ArrayList` would be returned by a call to `collectComments` with the given contents of `allReviews`. The reviews at index 1 and index 4 in `allReviews` are not included in the `ArrayList` to return since neither review contains an exclamation point.

"0-Good! Thx."	"2-Great!"	"3-Poor! Bad."
----------------	------------	----------------

Complete method `collectComments`.

```

/** Returns an ArrayList of String objects containing formatted versions of
 * selected user comments, as described in part (b)
 * Precondition: allReviews contains at least one Review.
 * No element of allReviews is null.
 * Postcondition: allReviews is unchanged.
 */
public ArrayList<String> collectComments()

```

Begin your response at the top of a new page in the Free Response booklet and fill in the appropriate circle indicating the question number.
If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

4. This question involves a two-dimensional array of integers that represents a collection of randomly generated data. A partial declaration of the `Data` class is shown. You will write two methods of the `Data` class.

```
public class Data
{
    public static final int MAX = /* value not shown */;
    private int[][] grid;

    /** Fills all elements of grid with randomly generated values, as described in part (a)
     * Precondition: grid is not null.
     * grid has at least one element.
     */
    public void repopulate()
    { /* to be implemented in part (a) */ }

    /** Returns the number of columns in grid that are in increasing order, as described
     * in part (b)
     * Precondition: grid is not null.
     * grid has at least one element.
     */
    public int countIncreasingCols()
    { /* to be implemented in part (b) */ }

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

GO ON TO THE NEXT PAGE.

- (a) Write the `repopulate` method, which assigns a newly generated random value to each element of `grid`. Each value is computed to meet all of the following criteria, and all valid values must have an equal chance of being generated.
- The value is between 1 and `MAX`, inclusive.
 - The value is divisible by 10.
 - The value is not divisible by 100.

Complete the `repopulate` method.

```
/** Fills all elements of grid with randomly generated values, as described in part (a)
 *   Precondition: grid is not null.
 *   grid has at least one element.
 */
public void repopulate()
```

**Begin your response at the top of a new page in the Free Response booklet
and fill in the appropriate circle indicating the question number.
If there are multiple parts to this question, write the part letter with your response.**

GO ON TO THE NEXT PAGE.

- (b) Write the `countIncreasingCols` method, which returns the number of columns in `grid` that are in increasing order. A column is considered to be in increasing order if the element in each row after the first row is greater than or equal to the element in the previous row. A column with only one row is considered to be in increasing order.

The following examples show the `countIncreasingCols` return values for possible contents of `grid`.

The return value for the following contents of `grid` is 1, since the first column is in increasing order but the second and third columns are not.

10	50	40
20	40	20
30	50	30

The return value for the following contents of `grid` is 2, since the first and third columns are in increasing order but the second and fourth columns are not.

10	540	440	440
220	450	440	190

Complete the `countIncreasingCols` method.

```
/** Returns the number of columns in grid that are in increasing order, as described
 * in part (b)
 * Precondition: grid is not null.
 * grid has at least one element.
 */
public int countIncreasingCols()
```

Begin your response at the top of a new page in the Free Response booklet and fill in the appropriate circle indicating the question number.
If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class Data
public static final int MAX = /* value not shown */
private int[][] grid
public void repopulate()
public int countIncreasingCols()
```

GO ON TO THE NEXT PAGE.



2021

AP[®]

CollegeBoard

AP[®] Computer Science A

Free-Response Questions



COMPUTER SCIENCE A

SECTION II

Time—1 hour and 30 minutes

4 Questions

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA. You may plan your answers in this orange booklet, but no credit will be given for anything written in this booklet. **You will only earn credit for what you write in the separate Free Response booklet.**

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

GO ON TO THE NEXT PAGE.

1. This question involves the `WordMatch` class, which stores a secret string and provides methods that compare other strings to the secret string. You will write two methods in the `WordMatch` class.

```
public class WordMatch
{
    /** The secret string. */
    private String secret;

    /** Constructs a WordMatch object with the given secret string of lowercase letters. */
    public WordMatch(String word)
    {
        /* implementation not shown */
    }

    /** Returns a score for guess, as described in part (a).
     *   Precondition: 0 < guess.length() <= secret.length()
     */
    public int scoreGuess(String guess)
    { /* to be implemented in part (a) */ }

    /** Returns the better of two guesses, as determined by scoreGuess and the rules for a
     *   tie-breaker that are described in part (b).
     *   Precondition: guess1 and guess2 contain all lowercase letters.
     *                   guess1 is not the same as guess2.
     */
    public String findBetterGuess(String guess1, String guess2)
    { /* to be implemented in part (b) */ }
}
```

GO ON TO THE NEXT PAGE.

- (a) Write the `WordMatch` method `scoreGuess`. To determine the score to be returned, `scoreGuess` finds the number of times that `guess` occurs as a substring of `secret` and then multiplies that number by the square of the length of `guess`. Occurrences of `guess` may overlap within `secret`.

Assume that the length of `guess` is less than or equal to the length of `secret` and that `guess` is not an empty string.

The following examples show declarations of a `WordMatch` object. The tables show the outcomes of some possible calls to the `scoreGuess` method.

```
WordMatch game = new WordMatch("mississippi");
```

Value of <code>guess</code>	Number of Substring Occurrences	Score Calculation: (Number of Substring Occurrences) x (Square of the Length of <code>guess</code>)	Return Value of <code>game.scoreGuess(guess)</code>
"i"	4	$4 * 1 * 1 = 4$	4
"iss"	2	$2 * 3 * 3 = 18$	18
"issipp"	1	$1 * 6 * 6 = 36$	36
"mississippi"	1	$1 * 11 * 11 = 121$	121

```
WordMatch game = new WordMatch("aaaabb");
```

Value of <code>guess</code>	Number of Substring Occurrences	Score Calculation: (Number of Substring Occurrences) x (Square of the Length of <code>guess</code>)	Return Value of <code>game.scoreGuess(guess)</code>
"a"	4	$4 * 1 * 1 = 4$	4
"aa"	3	$3 * 2 * 2 = 12$	12
"aaa"	2	$2 * 3 * 3 = 18$	18
"aabb"	1	$1 * 4 * 4 = 16$	16
"c"	0	$0 * 1 * 1 = 0$	0

GO ON TO THE NEXT PAGE.

Complete the `scoreGuess` method.

```
/** Returns a score for guess, as described in part (a).
 * Precondition: 0 < guess.length() <= secret.length()
 */
public int scoreGuess(String guess)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class WordMatch

private String secret

public WordMatch(String word)
public int scoreGuess(String guess)
public String findBetterGuess(String guess1, String guess2)
```

GO ON TO THE NEXT PAGE.

- (b) Write the `WordMatch` method `findBetterGuess`, which returns the better guess of its two `String` parameters, `guess1` and `guess2`. If the `scoreGuess` method returns different values for `guess1` and `guess2`, then the guess with the higher score is returned. If the `scoreGuess` method returns the same value for `guess1` and `guess2`, then the alphabetically greater guess is returned.

The following example shows a declaration of a `WordMatch` object and the outcomes of some possible calls to the `scoreGuess` and `findBetterGuess` methods.

```
WordMatch game = new WordMatch("concatenation");
```

Method Call	Return Value	Explanation
<code>game.scoreGuess("ten");</code>	9	1 * 3 * 3
<code>game.scoreGuess("nation");</code>	36	1 * 6 * 6
<code>game.findBetterGuess("ten", "nation");</code>	"nation"	Since <code>scoreGuess</code> returns 36 for "nation" and 9 for "ten", the guess with the greater score, "nation", is returned.
<code>game.scoreGuess("con");</code>	9	1 * 3 * 3
<code>game.scoreGuess("cat");</code>	9	1 * 3 * 3
<code>game.findBetterGuess("con", "cat");</code>	"con"	Since <code>scoreGuess</code> returns 9 for both "con" and "cat", the alphabetically greater guess, "con", is returned.

GO ON TO THE NEXT PAGE.

Complete method `findBetterGuess`.

Assume that `scoreGuess` works as specified, regardless of what you wrote in part (a). You must use `scoreGuess` appropriately to receive full credit.

```
/** Returns the better of two guesses, as determined by scoreGuess and the rules for a
 * tie-breaker that are described in part (b).
 * Precondition: guess1 and guess2 contain all lowercase letters.
 * guess1 is not the same as guess2.
 */
public String findBetterGuess(String guess1, String guess2)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class WordMatch

private String secret

public WordMatch(String word)
public int scoreGuess(String guess)
public String findBetterGuess(String guess1, String guess2)
```

GO ON TO THE NEXT PAGE.

2. The class `SingleTable` represents a table at a restaurant.

```
public class SingleTable
{
    /** Returns the number of seats at this table. The value is always greater than or equal to 4. */
    public int getNumSeats()
    { /* implementation not shown */ }

    /** Returns the height of this table in centimeters. */
    public int getHeight()
    { /* implementation not shown */ }

    /** Returns the quality of the view from this table. */
    public double getViewQuality()
    { /* implementation not shown */ }

    /** Sets the quality of the view from this table to value. */
    public void setViewQuality(double value)
    { /* implementation not shown */ }

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

At the restaurant, customers can sit at tables that are composed of two single tables pushed together. You will write a class `CombinedTable` to represent the result of combining two `SingleTable` objects, based on the following rules and the examples in the chart that follows.

- A `CombinedTable` can seat a number of customers that is two fewer than the total number of seats in its two `SingleTable` objects (to account for seats lost when the tables are pushed together).
- A `CombinedTable` has a desirability that depends on the views and heights of the two single tables. If the two single tables of a `CombinedTable` object are the same height, the desirability of the `CombinedTable` object is the average of the view qualities of the two single tables.
- If the two single tables of a `CombinedTable` object are not the same height, the desirability of the `CombinedTable` object is 10 units less than the average of the view qualities of the two single tables.

GO ON TO THE NEXT PAGE.

Assume `SingleTable` objects `t1`, `t2`, and `t3` have been created as follows.

- `SingleTable t1` has 4 seats, a view quality of 60.0, and a height of 74 centimeters.
- `SingleTable t2` has 8 seats, a view quality of 70.0, and a height of 74 centimeters.
- `SingleTable t3` has 12 seats, a view quality of 75.0, and a height of 76 centimeters.

The chart contains a sample code execution sequence and the corresponding results.

Statement	Value Returned (blank if no value)	Class Specification
<code>CombinedTable c1 = new CombinedTable(t1, t2);</code>		A <code>CombinedTable</code> is composed of two <code>SingleTable</code> objects.
<code>c1.canSeat(9);</code>	true	Since its two single tables have a total of 12 seats, <code>c1</code> can seat 10 or fewer people.
<code>c1.canSeat(11);</code>	false	<code>c1</code> cannot seat 11 people.
<code>c1.getDesirability();</code>	65.0	Because <code>c1</code> 's two single tables are the same height, its desirability is the average of 60.0 and 70.0.
<code>CombinedTable c2 = new CombinedTable(t2, t3);</code>		A <code>CombinedTable</code> is composed of two <code>SingleTable</code> objects.
<code>c2.canSeat(18);</code>	true	Since its two single tables have a total of 20 seats, <code>c2</code> can seat 18 or fewer people.
<code>c2.getDesirability();</code>	62.5	Because <code>c2</code> 's two single tables are not the same height, its desirability is 10 units less than the average of 70.0 and 75.0.
<code>t2.setViewQuality(80);</code>		Changing the view quality of one of the tables that makes up <code>c2</code> changes the desirability of <code>c2</code> , as illustrated in the next line of the chart. Since <code>setViewQuality</code> is a <code>SingleTable</code> method, you do not need to write it.
<code>c2.getDesirability();</code>	67.5	Because the view quality of <code>t2</code> changed, the desirability of <code>c2</code> has also changed.

The last line of the chart illustrates that when the characteristics of a `SingleTable` change, so do those of the `CombinedTable` that contains it.

Write the complete `CombinedTable` class. Your implementation must meet all specifications and conform to the examples shown in the preceding chart.

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

3. A high school club maintains information about its members in a `MemberInfo` object. A `MemberInfo` object stores a club member's name, year of graduation, and whether or not the club member is in *good standing*. A member who is in good standing has fulfilled all the responsibilities of club membership.

A partial declaration of the `MemberInfo` class is shown below.

```
public class MemberInfo
{
    /** Constructs a MemberInfo object for the club member with name name,
     * graduation year gradYear, and standing hasGoodStanding.
     */
    public MemberInfo(String name, int gradYear, boolean hasGoodStanding)
    { /* implementation not shown */ }

    /** Returns the graduation year of the club member. */
    public int getGradYear()
    { /* implementation not shown */ }

    /** Returns true if the member is in good standing and false otherwise. */
    public boolean inGoodStanding()
    { /* implementation not shown */ }

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

The `ClubMembers` class maintains a list of current club members. The declaration of the `ClubMembers` class is shown below.

```
public class ClubMembers
{
    private ArrayList<MemberInfo> memberList;

    /** Adds new club members to memberList, as described in part (a).
     * Precondition: names is a non-empty array.
     */
    public void addMembers(String[] names, int gradYear)
    { /* to be implemented in part (a) */ }

    /** Removes members who have graduated and returns a list of members who have graduated
     * and are in good standing, as described in part (b).
     */
    public ArrayList<MemberInfo> removeMembers(int year)
    { /* to be implemented in part (b) */ }

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

GO ON TO THE NEXT PAGE.

- (a) Write the `ClubMembers` method `addMembers`, which takes two parameters. The first parameter is a `String` array containing the names of new club members to be added. The second parameter is the graduation year of all the new club members. The method adds the new members to the `memberList` instance variable. The names can be added in any order. All members added are initially in good standing and share the same graduation year, `gradYear`.

Complete the `addMembers` method.

```
/** Adds new club members to memberList, as described in part (a).
 * Precondition: names is a non-empty array.
 */
public void addMembers(String[] names, int gradYear)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

(b) Write the `ClubMembers` method `removeMembers`, which takes the following actions.

- Returns a list of all students who have graduated and are in good standing. A member has graduated if the member's graduation year is less than or equal to the method's `year` parameter. If no members meet these criteria, an empty list is returned.
- Removes from `memberList` all members who have graduated, regardless of whether or not they are in good standing.

The following example illustrates the results of a call to `removeMembers`.

The `ArrayList` `memberList` before the method call `removeMembers(2018)`:

"SMITH, JANE"	"FOX, STEVE"	"XIN, MICHAEL"	"GARCIA, MARIA"
2019	2018	2017	2020
false	true	false	true

The `ArrayList` `memberList` after the method call `removeMembers(2018)`:

"SMITH, JANE"	"GARCIA, MARIA"
2019	2020
false	true

The `ArrayList` returned by the method call `removeMembers(2018)`:

"FOX, STEVE"
2018
true

GO ON TO THE NEXT PAGE.

Complete the `removeMembers` method.

```
/** Removes members who have graduated and returns a list of members who have graduated and are
 *   in good standing, as described in part (b).
 */
public ArrayList<MemberInfo> removeMembers(int year)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class MemberInfo
```

```
public MemberInfo(String name, int gradYear, boolean hasGoodStanding)
public int getGradYear()
public boolean inGoodStanding()
```

```
public class ClubMembers
```

```
private ArrayList<MemberInfo> memberList
```

```
public void addMembers(String[] names, int gradYear)
public ArrayList<MemberInfo> removeMembers(int year)
```

GO ON TO THE NEXT PAGE.

4. This question involves manipulating a two-dimensional array of integers. You will write two static methods of the `ArrayResizer` class, which is shown below.

```
public class ArrayResizer
{
    /** Returns true if and only if every value in row r of array2D is non-zero.
     * Precondition: r is a valid row index in array2D.
     * Postcondition: array2D is unchanged.
     */
    public static boolean isNonZeroRow(int[][] array2D, int r)
    { /* to be implemented in part (a) */ }

    /** Returns the number of rows in array2D that contain all non-zero values.
     * Postcondition: array2D is unchanged.
     */
    public static int numNonZeroRows(int[][] array2D)
    { /* implementation not shown */ }

    /** Returns a new, possibly smaller, two-dimensional array that contains only rows
     * from array2D with no zeros, as described in part (b).
     * Precondition: array2D contains at least one column and at least one row with no zeros.
     * Postcondition: array2D is unchanged.
     */
    public static int[][] resize(int[][] array2D)
    { /* to be implemented in part (b) */ }
}
```

GO ON TO THE NEXT PAGE.

- (a) Write the method `isNonZeroRow`, which returns `true` if and only if all elements in row `r` of a two-dimensional array `array2D` are not equal to zero.

For example, consider the following statement, which initializes a two-dimensional array.

```
int[][] arr = {{2, 1, 0},
               {1, 3, 2},
               {0, 0, 0},
               {4, 5, 6}};
```

Sample calls to `isNonZeroRow` are shown below.

Call to <code>isNonZeroRow</code>	Value Returned	Explanation
<code>ArrayResizer.isNonZeroRow(arr, 0)</code>	<code>false</code>	At least one value in row 0 is zero.
<code>ArrayResizer.isNonZeroRow(arr, 1)</code>	<code>true</code>	All values in row 1 are non-zero.
<code>ArrayResizer.isNonZeroRow(arr, 2)</code>	<code>false</code>	At least one value in row 2 is zero.
<code>ArrayResizer.isNonZeroRow(arr, 3)</code>	<code>true</code>	All values in row 3 are non-zero.

Complete the `isNonZeroRow` method.

```
/** Returns true if and only if every value in row r of array2D is non-zero.
 *   Precondition: r is a valid row index in array2D.
 *   Postcondition: array2D is unchanged.
 */
public static boolean isNonZeroRow(int[][] array2D, int r)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

- (b) Write the method `resize`, which returns a new two-dimensional array containing only rows from `array2D` with all non-zero values. The elements in the new array should appear in the same order as the order in which they appeared in the original array.

The following code segment initializes a two-dimensional array and calls the `resize` method.

```
int[][] arr = {{2, 1, 0},
               {1, 3, 2},
               {0, 0, 0},
               {4, 5, 6}};
int[][] smaller = ArrayResizer.resize(arr);
```

When the code segment completes, the following will be the contents of `smaller`.

```
{{1, 3, 2}, {4, 5, 6}}
```

A helper method, `numNonZeroRows`, has been provided for you. The method returns the number of rows in its two-dimensional array parameter that contain no zero values.

Complete the `resize` method. Assume that `isNonZeroRow` works as specified, regardless of what you wrote in part (a). You must use `numNonZeroRows` and `isNonZeroRow` appropriately to receive full credit.

```
/** Returns a new, possibly smaller, two-dimensional array that contains only rows from array2D
 * with no zeros, as described in part (b).
 * Precondition: array2D contains at least one column and at least one row with no zeros.
 * Postcondition: array2D is unchanged.
 */
public static int[][] resize(int[][] array2D)
```

Begin your response at the top of a new page in the separate Free Response booklet and fill in the appropriate circle at the top of each page to indicate the question number. If there are multiple parts to this question, write the part letter with your response.

Class information for this question

```
public class ArrayResizer

public static boolean isNonZeroRow(int[][] array2D, int r)
public static int numNonZeroRows(int[][] array2D)
public static int[][] resize(int[][] array2D)
```

GO ON TO THE NEXT PAGE.



2019

AP[®]

CollegeBoard

AP[®] Computer Science A

Free-Response Questions



© 2019 The College Board. College Board, Advanced Placement, AP, AP Central, and the acorn logo are registered trademarks of the College Board. Visit the College Board on the web: collegeboard.org.

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

2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

COMPUTER SCIENCE A SECTION II

Time—1 hour and 30 minutes

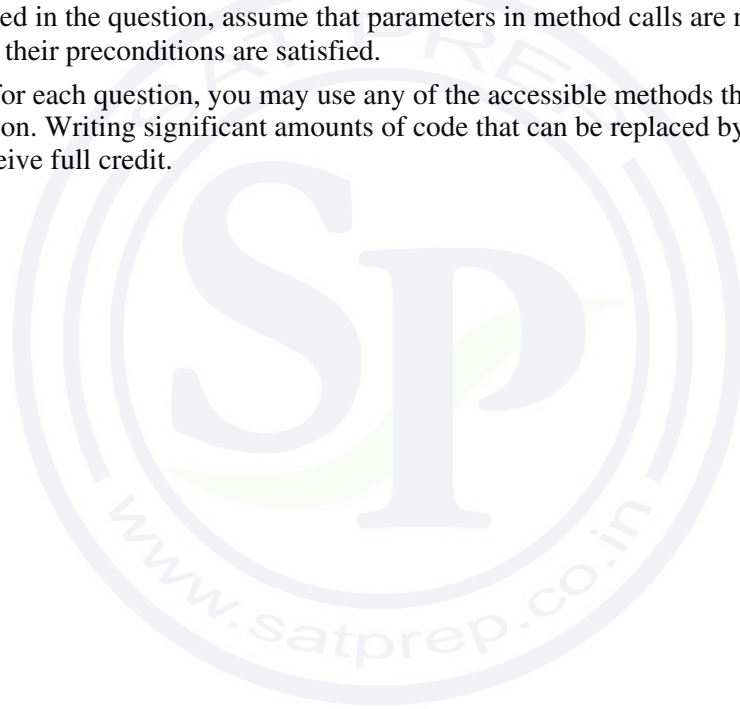
Number of questions—4

Percent of total score—50

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the interface and classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.



2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

1. The `APCalendar` class contains methods used to calculate information about a calendar. You will write two methods of the class.

```
public class APCalendar
{

    /** Returns true if year is a leap year and false otherwise. */
    private static boolean isLeapYear(int year)
    { /* implementation not shown */ }

    /** Returns the number of leap years between year1 and year2, inclusive.
     *   Precondition: 0 <= year1 <= year2
     */
    public static int numberOfLeapYears(int year1, int year2)
    { /* to be implemented in part (a) */ }

    /** Returns the value representing the day of the week for the first day of year,
     *   where 0 denotes Sunday, 1 denotes Monday, ..., and 6 denotes Saturday.
     */
    private static int firstDayOfYear(int year)
    { /* implementation not shown */ }

    /** Returns n, where month, day, and year specify the nth day of the year.
     *   Returns 1 for January 1 (month = 1, day = 1) of any year.
     *   Precondition: The date represented by month, day, year is a valid date.
     */
    private static int dayOfYear(int month, int day, int year)
    { /* implementation not shown */ }

    /** Returns the value representing the day of the week for the given date
     *   (month, day, year), where 0 denotes Sunday, 1 denotes Monday, ...,
     *   and 6 denotes Saturday.
     *   Precondition: The date represented by month, day, year is a valid date.
     */
    public static int dayOfWeek(int month, int day, int year)
    { /* to be implemented in part (b) */ }

    // There may be instance variables, constructors, and other methods not shown.

}
```

2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the static method `numberOfLeapYears`, which returns the number of leap years between `year1` and `year2`, inclusive.

In order to calculate this value, a helper method is provided for you.

- `isLeapYear(year)` returns `true` if `year` is a leap year and `false` otherwise.

Complete method `numberOfLeapYears` below. You must use `isLeapYear` appropriately to receive full credit.

```
/** Returns the number of leap years between year1 and year2, inclusive.
 *   Precondition: 0 <= year1 <= year2
 */
public static int numberOfLeapYears(int year1, int year2)
```



2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the static method `dayOfWeek`, which returns the integer value representing the day of the week for the given date (month, day, year), where 0 denotes Sunday, 1 denotes Monday, ..., and 6 denotes Saturday. For example, 2019 began on a Tuesday, and January 5 is the fifth day of 2019. As a result, January 5, 2019, fell on a Saturday, and the method call `dayOfWeek(1, 5, 2019)` returns 6.

As another example, January 10 is the tenth day of 2019. As a result, January 10, 2019, fell on a Thursday, and the method call `dayOfWeek(1, 10, 2019)` returns 4.

In order to calculate this value, two helper methods are provided for you.

- `firstDayOfYear(year)` returns the integer value representing the day of the week for the first day of year, where 0 denotes Sunday, 1 denotes Monday, ..., and 6 denotes Saturday. For example, since 2019 began on a Tuesday, `firstDayOfYear(2019)` returns 2.
- `dayOfYear(month, day, year)` returns n , where month, day, and year specify the n th day of the year. For the first day of the year, January 1 (month = 1, day = 1), the value 1 is returned. This method accounts for whether year is a leap year. For example, `dayOfYear(3, 1, 2017)` returns 60, since 2017 is not a leap year, while `dayOfYear(3, 1, 2016)` returns 61, since 2016 is a leap year.

Class information for this question

```
public class APCalendar
```

```
private static boolean isLeapYear(int year)
public static int numberOfLeapYears(int year1, int year2)
private static int firstDayOfYear(int year)
private static int dayOfYear(int month, int day, int year)
public static int dayOfWeek(int month, int day, int year)
```

2019 AP® COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `dayOfWeek` below. You must use `firstDayOfYear` and `dayOfYear` appropriately to receive full credit.

```
/** Returns the value representing the day of the week for the given date
 * (month, day, year), where 0 denotes Sunday, 1 denotes Monday, ...,
 * and 6 denotes Saturday.
 * Precondition: The date represented by month, day, year is a valid date.
 */
public static int dayOfWeek(int month, int day, int year)
```



2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

2. This question involves the implementation of a fitness tracking system that is represented by the `StepTracker` class. A `StepTracker` object is created with a parameter that defines the minimum number of steps that must be taken for a day to be considered *active*.

The `StepTracker` class provides a constructor and the following methods.

- `addDailySteps`, which accumulates information about steps, in readings taken once per day
- `activeDays`, which returns the number of active days
- `averageSteps`, which returns the average number of steps per day, calculated by dividing the total number of steps taken by the number of days tracked

The following table contains a sample code execution sequence and the corresponding results.

Statements and Expressions	Value Returned (blank if no value)	Comment
<code>StepTracker tr = new StepTracker(10000);</code>		Days with at least 10,000 steps are considered active. Assume that the parameter is positive.
<code>tr.activeDays();</code>	0	No data have been recorded yet.
<code>tr.averageSteps();</code>	0.0	When no step data have been recorded, the <code>averageSteps</code> method returns 0.0.
<code>tr.addDailySteps(9000);</code>		This is too few steps for the day to be considered active.
<code>tr.addDailySteps(5000);</code>		This is too few steps for the day to be considered active.
<code>tr.activeDays();</code>	0	No day had at least 10,000 steps.
<code>tr.averageSteps();</code>	7000.0	The average number of steps per day is (14000 / 2).
<code>tr.addDailySteps(13000);</code>		This represents an active day.
<code>tr.activeDays();</code>	1	Of the three days for which step data were entered, one day had at least 10,000 steps.
<code>tr.averageSteps();</code>	9000.0	The average number of steps per day is (27000 / 3).
<code>tr.addDailySteps(23000);</code>		This represents an active day.
<code>tr.addDailySteps(1111);</code>		This is too few steps for the day to be considered active.
<code>tr.activeDays();</code>	2	Of the five days for which step data were entered, two days had at least 10,000 steps.
<code>tr.averageSteps();</code>	10222.2	The average number of steps per day is (51111 / 5).

2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Write the complete `StepTracker` class, including the constructor and any required instance variables and methods. Your implementation must meet all specifications and conform to the example.



2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

3. Many encoded strings contain *delimiters*. A delimiter is a non-empty string that acts as a boundary between different parts of a larger string. The delimiters involved in this question occur in pairs that must be *balanced*, with each pair having an open delimiter and a close delimiter. There will be only one type of delimiter for each string. The following are examples of delimiters.

Example 1

Expressions in mathematics use open parentheses " (" and close parentheses ") " as delimiters. For each open parenthesis, there must be a matching close parenthesis.

$(x + y) * 5$ is a valid mathematical expression.

$(x + (y)$ is NOT a valid mathematical expression because there are more open delimiters than close delimiters.

Example 2

HTML uses `` and `` as delimiters. For each open delimiter ``, there must be a matching close delimiter ``.

` Make this text bold ` is valid HTML.

` Make this text bold </UB>` is NOT valid HTML because there is one open delimiter and no matching close delimiter.

2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

In this question, you will write two methods in the following `Delimiters` class.

```
public class Delimiters
{
    /** The open and close delimiters. */
    private String openDel;
    private String closeDel;

    /** Constructs a Delimiters object where open is the open delimiter and close is the
     * close delimiter.
     * Precondition: open and close are non-empty strings.
     */
    public Delimiters(String open, String close)
    {
        openDel = open;
        closeDel = close;
    }

    /** Returns an ArrayList of delimiters from the array tokens, as described in part (a). */
    public ArrayList<String> getDelimitersList(String[] tokens)
    {
        /* to be implemented in part (a) */
    }

    /** Returns true if the delimiters are balanced and false otherwise, as described in part (b).
     * Precondition: delimiters contains only valid open and close delimiters.
     */
    public boolean isBalanced(ArrayList<String> delimiters)
    {
        /* to be implemented in part (b) */
    }

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

2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) A string containing text and possibly delimiters has been split into *tokens* and stored in `String[] tokens`. Each token is either an open delimiter, a close delimiter, or a substring that is not a delimiter. You will write the method `getDelimitersList`, which returns an `ArrayList` containing all the open and close delimiters found in `tokens` in their original order.

The following examples show the contents of an `ArrayList` returned by `getDelimitersList` for different open and close delimiters and different `tokens` arrays.

Example 1

openDel: "("

closeDel: ")"

tokens:

"("	"x + y"	")"	" * 5"
-----	---------	-----	--------

ArrayList
of delimiters:

"("	")"
-----	-----

Example 2

openDel: "<q>"

closeDel: "</q>"

tokens:

"<q>"	"yy"	"</q>"	"zz"	"</q>"
-------	------	--------	------	--------

ArrayList
of delimiters:

"<q>"	"</q>"	"</q>"
-------	--------	--------

Class information for this question

```
public class Delimiters
private String openDel
private String closeDel

public Delimiters(String open, String close)
public ArrayList<String> getDelimitersList(String[] tokens)
public boolean isBalanced(ArrayList<String> delimiters)
```

2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `getDelimitersList` below.

```
/** Returns an ArrayList of delimiters from the array tokens, as described in part (a). */  
public ArrayList<String> getDelimitersList(String[] tokens)
```



2019 AP® COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the method `isBalanced`, which returns `true` when the delimiters are balanced and returns `false` otherwise. The delimiters are balanced when both of the following conditions are satisfied; otherwise, they are not balanced.
1. When traversing the `ArrayList` from the first element to the last element, there is no point at which there are more close delimiters than open delimiters at or before that point.
 2. The total number of open delimiters is equal to the total number of close delimiters.

Consider a `Delimiters` object for which `openDel` is `"^{"` and `closeDel` is `"}"`. The examples below show different `ArrayList` objects that could be returned by calls to `getDelimitersList` and the value that would be returned by a call to `isBalanced`.

Example 1

The following example shows an `ArrayList` for which `isBalanced` returns `true`. As tokens are examined from first to last, the number of open delimiters is always greater than or equal to the number of close delimiters. After examining all tokens, there are an equal number of open and close delimiters.

"^{"	"^{"	"}"	"^{"	"}"	"}"
---------	---------	----------	---------	----------	----------

Example 2

The following example shows an `ArrayList` for which `isBalanced` returns `false`.

"^{"	"}"	"</sup>"	"<sup>"
---------	----------	----------	---------



When starting from the left, at this point, condition 1 is violated.

Example 3

The following example shows an `ArrayList` for which `isBalanced` returns `false`.

"</sup>"



At this point, condition 1 is violated.

Example 4

The following example shows an `ArrayList` for which `isBalanced` returns `false` because the second condition is violated. After examining all tokens, there are not an equal number of open and close delimiters.

"<sup>"	"^{"	"}"
---------	---------	----------

2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Class information for this question

```
public class Delimiters  
private String openDel  
private String closeDel  
  
public Delimiters(String open, String close)  
public ArrayList<String> getDelimitersList(String[] tokens)  
public boolean isBalanced(ArrayList<String> delimiters)
```

Complete method `isBalanced` below.

```
/** Returns true if the delimiters are balanced and false otherwise, as described in part (b).  
 * Precondition: delimiters contains only valid open and close delimiters.  
 */  
public boolean isBalanced(ArrayList<String> delimiters)
```



2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

4. The `LightBoard` class models a two-dimensional display of lights, where each light is either on or off, as represented by a Boolean value. You will implement a constructor to initialize the display and a method to evaluate a light.

```
public class LightBoard
{
    /** The lights on the board, where true represents on and false represents off.
     */
    private boolean[][] lights;

    /** Constructs a LightBoard object having numRows rows and numCols columns.
     * Precondition: numRows > 0, numCols > 0
     * Postcondition: each light has a 40% probability of being set to on.
     */
    public LightBoard(int numRows, int numCols)
    { /* to be implemented in part (a) */ }

    /** Evaluates a light in row index row and column index col and returns a status
     * as described in part (b).
     * Precondition: row and col are valid indexes in lights.
     */
    public boolean evaluateLight(int row, int col)
    { /* to be implemented in part (b) */ }

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

2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the constructor for the `LightBoard` class, which initializes `lights` so that each light is set to on with a 40% probability. The notation `lights[r][c]` represents the array element at row `r` and column `c`.

Complete the `LightBoard` constructor below.

```
/** Constructs a LightBoard object having numRows rows and numCols columns.
 *   Precondition: numRows > 0, numCols > 0
 *   Postcondition: each light has a 40% probability of being set to on.
 */
public LightBoard(int numRows, int numCols)
```



2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the method `evaluateLight`, which computes and returns the status of a light at a given row and column based on the following rules.
1. If the light is on, return `false` if the number of lights in its column that are on is even, including the current light.
 2. If the light is off, return `true` if the number of lights in its column that are on is divisible by three.
 3. Otherwise, return the light's current status.

For example, suppose that `LightBoard sim = new LightBoard(7, 5)` creates a light board with the initial state shown below, where `true` represents a light that is on and `false` represents a light that is off. Lights that are off are shaded.

lights

	0	1	2	3	4
0	true	true	false	true	true
1	true	false	false	true	false
2	true	false	false	true	true
3	true	false	false	false	true
4	true	false	false	false	true
5	true	true	false	true	true
6	false	false	false	false	false

Sample calls to `evaluateLight` are shown below.

Call to <code>evaluateLight</code>	Value Returned	Explanation
<code>sim.evaluateLight(0, 3);</code>	<code>false</code>	The light is on, and the number of lights that are on in its column is even.
<code>sim.evaluateLight(6, 0);</code>	<code>true</code>	The light is off, and the number of lights that are on in its column is divisible by 3.
<code>sim.evaluateLight(4, 1);</code>	<code>false</code>	Returns the light's current status.
<code>sim.evaluateLight(5, 4);</code>	<code>true</code>	Returns the light's current status.

2019 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Class information for this question

```
public class LightBoard
private boolean[][] lights
public LightBoard(int numRows, int numCols)
public boolean evaluateLight(int row, int col)
```

Complete the `evaluateLight` method below.

```
/** Evaluates a light in row index row and column index col and returns a status
 * as described in part (b).
 * Precondition: row and col are valid indexes in lights.
 */
public boolean evaluateLight(int row, int col)
```



STOP

END OF EXAM

2018

AP[®]

CollegeBoard

AP Computer Science A

Free-Response Questions



© 2018 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.

2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

COMPUTER SCIENCE A SECTION II

Time—1 hour and 30 minutes

Number of questions—4

Percent of total score—50

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the interface and classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.



2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

1. This question involves reasoning about a simulation of a frog hopping in a straight line. The frog attempts to hop to a goal within a specified number of hops. The simulation is encapsulated in the following `FrogSimulation` class. You will write two of the methods in this class.

```
public class FrogSimulation
{
    /** Distance, in inches, from the starting position to the goal. */
    private int goalDistance;

    /** Maximum number of hops allowed to reach the goal. */
    private int maxHops;

    /** Constructs a FrogSimulation where dist is the distance, in inches, from the starting
     * position to the goal, and numHops is the maximum number of hops allowed to reach the goal.
     * Precondition: dist > 0; numHops > 0
     */
    public FrogSimulation(int dist, int numHops)
    {
        goalDistance = dist;
        maxHops = numHops;
    }

    /** Returns an integer representing the distance, in inches, to be moved when the frog hops.
     */
    private int hopDistance()
    {
        /* implementation not shown */
    }

    /** Simulates a frog attempting to reach the goal as described in part (a).
     * Returns true if the frog successfully reached or passed the goal during the simulation;
     * false otherwise.
     */
    public boolean simulate()
    {
        /* to be implemented in part (a) */
    }

    /** Runs num simulations and returns the proportion of simulations in which the frog
     * successfully reached or passed the goal.
     * Precondition: num > 0
     */
    public double runSimulations(int num)
    {
        /* to be implemented in part (b) */
    }
}
```

2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the `simulate` method, which simulates the frog attempting to hop in a straight line to a goal from the frog's starting position of 0 within a maximum number of hops. The method returns `true` if the frog successfully reached the goal within the maximum number of hops; otherwise, the method returns `false`.

The `FrogSimulation` class provides a method called `hopDistance` that returns an integer representing the distance (positive or negative) to be moved when the frog hops. A positive distance represents a move toward the goal. A negative distance represents a move away from the goal. The returned distance may vary from call to call. Each time the frog hops, its position is adjusted by the value returned by a call to the `hopDistance` method.

The frog hops until one of the following conditions becomes true:

- The frog has reached or passed the goal.
- The frog has reached a negative position.
- The frog has taken the maximum number of hops without reaching the goal.

The following example shows a declaration of a `FrogSimulation` object for which the goal distance is 24 inches and the maximum number of hops is 5. The table shows some possible outcomes of calling the `simulate` method.

```
FrogSimulation sim = new FrogSimulation(24, 5);
```

	Values returned by <code>hopDistance()</code>	Final position of frog	Return value of <code>sim.simulate()</code>
Example 1	5, 7, -2, 8, 6	24	true
Example 2	6, 7, 6, 6	25	true
Example 3	6, -6, 31	31	true
Example 4	4, 2, -8	-2	false
Example 5	5, 4, 2, 4, 3	18	false

Class information for this question

```
public class FrogSimulation  
  
private int goalDistance  
private int maxHops  
  
private int hopDistance()  
public boolean simulate()  
public double runSimulations(int num)
```

2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `simulate` below. You must use `hopDistance` appropriately to receive full credit.

```
/** Simulates a frog attempting to reach the goal as described in part (a).
 * Returns true if the frog successfully reached or passed the goal during the simulation;
 *         false otherwise.
 */
public boolean simulate()
```



2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `runSimulations` method, which performs a given number of simulations and returns the proportion of simulations in which the frog successfully reached or passed the goal. For example, if the parameter passed to `runSimulations` is 400, and 100 of the 400 `simulate` method calls returned `true`, then the `runSimulations` method should return 0.25.

Complete method `runSimulations` below. Assume that `simulate` works as specified, regardless of what you wrote in part (a). You must use `simulate` appropriately to receive full credit.

```
/** Runs num simulations and returns the proportion of simulations in which the frog
 * successfully reached or passed the goal.
 * Precondition: num > 0
 */
public double runSimulations(int num)
```



2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

2. This question involves reasoning about pairs of words that are represented by the following `WordPair` class.

```
public class WordPair
{
    /** Constructs a WordPair object. */
    public WordPair(String first, String second)
    { /* implementation not shown */ }

    /** Returns the first string of this WordPair object. */
    public String getFirst()
    { /* implementation not shown */ }

    /** Returns the second string of this WordPair object. */
    public String getSecond()
    { /* implementation not shown */ }
}
```

You will implement the constructor and another method for the following `WordPairList` class.

```
public class WordPairList
{
    /** The list of word pairs, initialized by the constructor. */
    private ArrayList<WordPair> allPairs;

    /** Constructs a WordPairList object as described in part (a).
     *   Precondition: words.length >= 2
     */
    public WordPairList(String[] words)
    { /* to be implemented in part (a) */ }

    /** Returns the number of matches as described in part (b).
     */
    public int numMatches()
    { /* to be implemented in part (b) */ }
}
```

2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the constructor for the `WordPairList` class. The constructor takes an array of strings `words` as a parameter and initializes the instance variable `allPairs` to an `ArrayList` of `WordPair` objects.

A `WordPair` object consists of a word from the array paired with a word that appears later in the array. The `allPairs` list contains `WordPair` objects (`words[i]`, `words[j]`) for every `i` and `j`, where $0 \leq i < j < \text{words.length}$. Each `WordPair` object is added exactly once to the list.

The following examples illustrate two different `WordPairList` objects.

Example 1

```
String[] wordNums = {"one", "two", "three"};
WordPairList exampleOne = new WordPairList(wordNums);
```

After the code segment has executed, the `allPairs` instance variable of `exampleOne` will contain the following `WordPair` objects in some order.

```
("one", "two"), ("one", "three"), ("two", "three")
```

Example 2

```
String[] phrase = {"the", "more", "the", "merrier"};
WordPairList exampleTwo = new WordPairList(phrase);
```

After the code segment has executed, the `allPairs` instance variable of `exampleTwo` will contain the following `WordPair` objects in some order.

```
("the", "more"), ("the", "the"), ("the", "merrier"),
("more", "the"), ("more", "merrier"), ("the", "merrier")
```

Class information for this question

```
public class WordPair
```

```
public WordPair(String first, String second)
public String getFirst()
public String getSecond()
```

```
public class WordPairList
```

```
private ArrayList<WordPair> allPairs

public WordPairList(String[] words)
public int numMatches()
```

2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete the `WordPairList` constructor below.

```
/** Constructs a WordPairList object as described in part (a).  
 *   Precondition: words.length >= 2  
 */  
public WordPairList(String[] words)
```



2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `WordPairList` method `numMatches`. This method returns the number of `WordPair` objects in `allPairs` for which the two strings match.

For example, the following code segment creates a `WordPairList` object.

```
String[] moreWords = {"the", "red", "fox", "the", "red"};
WordPairList exampleThree = new WordPairList(moreWords);
```

After the code segment has executed, the `allPairs` instance variable of `exampleThree` will contain the following `WordPair` objects in some order. The pairs in which the first string matches the second string are shaded for illustration.

```
("the", "red"), ("the", "fox"), ("the", "the"),
("the", "red"), ("red", "fox"), ("red", "the"),
("red", "red"), ("fox", "the"), ("fox", "red"),
("the", "red")
```

The call `exampleThree.numMatches()` should return 2.

Class information for this question

```
public class WordPair
```

```
public WordPair(String first, String second)
public String getFirst()
public String getSecond()
```

```
public class WordPairList
```

```
private ArrayList<WordPair> allPairs
```

```
public WordPairList(String[] words)
public int numMatches()
```

2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `numMatches` below.

```
/** Returns the number of matches as described in part (b).  
 */  
public int numMatches()
```



2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

3. The `StringChecker` interface describes classes that check if strings are valid, according to some criterion.

```
public interface StringChecker
{
    /** Returns true if str is valid. */
    boolean isValid(String str);
}
```

A `CodeWordChecker` is a `StringChecker`. A `CodeWordChecker` object can be constructed with three parameters: two integers and a string. The first two parameters specify the minimum and maximum code word lengths, respectively, and the third parameter specifies a string that must not occur in the code word. A `CodeWordChecker` object can also be constructed with a single parameter that specifies a string that must not occur in the code word; in this case the minimum and maximum lengths will default to 6 and 20, respectively.

The following examples illustrate the behavior of `CodeWordChecker` objects.

Example 1

```
StringChecker sc1 = new CodeWordChecker(5, 8, "$");
```

Valid code words have 5 to 8 characters and must not include the string "\$".

Method call	Return value	Explanation
<code>sc1.isValid("happy")</code>	true	The code word is valid.
<code>sc1.isValid("happy\$")</code>	false	The code word contains "\$".
<code>sc1.isValid("Code")</code>	false	The code word is too short.
<code>sc1.isValid("happyCode")</code>	false	The code word is too long.

Example 2

```
StringChecker sc2 = new CodeWordChecker("pass");
```

Valid code words must not include the string "pass". Because the bounds are not specified, the length bounds are 6 and 20, inclusive.

Method call	Return value	Explanation
<code>sc2.isValid("MyPass")</code>	true	The code word is valid.
<code>sc2.isValid("Mypassport")</code>	false	The code word contains "pass".
<code>sc2.isValid("happy")</code>	false	The code word is too short.
<code>sc2.isValid("1,000,000,000,000,000")</code>	false	The code word is too long.

2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Write the complete `CodeWordChecker` class. Your implementation must meet all specifications and conform to all examples.



2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

4. This question involves reasoning about arrays of integers. You will write two static methods, both of which are in a class named `ArrayTester`.

```
public class ArrayTester
{
    /** Returns an array containing the elements of column c of arr2D in the same order as
     * they appear in arr2D.
     * Precondition: c is a valid column index in arr2D.
     * Postcondition: arr2D is unchanged.
     */
    public static int[] getColumn(int[][] arr2D, int c)
    { /* to be implemented in part (a) */ }

    /** Returns true if and only if every value in arr1 appears in arr2.
     * Precondition: arr1 and arr2 have the same length.
     * Postcondition: arr1 and arr2 are unchanged.
     */
    public static boolean hasAllValues(int[] arr1, int[] arr2)
    { /* implementation not shown */ }

    /** Returns true if arr contains any duplicate values;
     * false otherwise.
     */
    public static boolean containsDuplicates(int[] arr)
    { /* implementation not shown */ }

    /** Returns true if square is a Latin square as described in part (b);
     * false otherwise.
     * Precondition: square has an equal number of rows and columns.
     * square has at least one row.
     */
    public static boolean isLatin(int[][] square)
    { /* to be implemented in part (b) */ }
}
```


2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write a static method `getColumn`, which returns a one-dimensional array containing the elements of a single column in a two-dimensional array. The elements in the returned array should be in the same order as they appear in the given column. The notation `arr2D[r][c]` represents the array element at row `r` and column `c`.

The following code segment initializes an array and calls the `getColumn` method.

```
int[][] arr2D = { { 0, 1, 2 },  
                  { 3, 4, 5 },  
                  { 6, 7, 8 },  
                  { 9, 5, 3 } };  
  
int[] result = ArrayTester.getColumn(arr2D, 1);
```

When the code segment has completed execution, the variable `result` will have the following contents.

`result: {1, 4, 7, 5}`

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `getColumn` below.

```
/** Returns an array containing the elements of column c of arr2D in the same order as they
 * appear in arr2D.
 * Precondition: c is a valid column index in arr2D.
 * Postcondition: arr2D is unchanged.
 */
public static int[] getColumn(int[][] arr2D, int c)
```



2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the static method `isLatin`, which returns `true` if a given two-dimensional square array is a *Latin square*, and otherwise, returns `false`.

A two-dimensional square array of integers is a Latin square if the following conditions are true.

- The first row has no duplicate values.
- All values in the first row of the square appear in each row of the square.
- All values in the first row of the square appear in each column of the square.

Examples of Latin Squares

1	2	3
2	3	1
3	1	2

10	30	20	0
0	20	30	10
30	0	10	20
20	10	0	30

Examples that are NOT Latin Squares

1	2	1
2	1	1
1	1	2

Not a Latin square
because the first row
contains duplicate
values

1	2	3
3	1	2
7	8	9

Not a Latin square
because the elements of
the first row do not all
appear in the third row

1	2
1	2

Not a Latin square
because the elements of
the first row do not all
appear in either column

The `ArrayTester` class provides two helper methods: `containsDuplicates` and `hasAllValues`. The method `containsDuplicates` returns `true` if the given one-dimensional array `arr` contains any duplicate values and `false` otherwise. The method `hasAllValues` returns `true` if and only if every value in `arr1` appears in `arr2`. You do not need to write the code for these methods.

Class information for this question

```
public class ArrayTester
```

```
public static int[] getColumn(int[][] arr2D, int c)
public static boolean hasAllValues(int[] arr1, int[] arr2)
public static boolean containsDuplicates(int[] arr)
public static boolean isLatin(int[][] square)
```

2018 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `isLatin` below. Assume that `getColumn` works as specified, regardless of what you wrote in part (a). You must use `getColumn`, `hasAllValues`, and `containsDuplicates` appropriately to receive full credit.

```
/** Returns true if square is a Latin square as described in part (b);
 *     false otherwise.
 * Precondition: square has an equal number of rows and columns.
 *                 square has at least one row.
 */
public static boolean isLatin(int[][] square)
```



STOP

END OF EXAM

2017

AP[®]

CollegeBoard

AP Computer Science A

Free-Response Questions



© 2017 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.

2017 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

**COMPUTER SCIENCE A
SECTION II**

Time—1 hour and 30 minutes

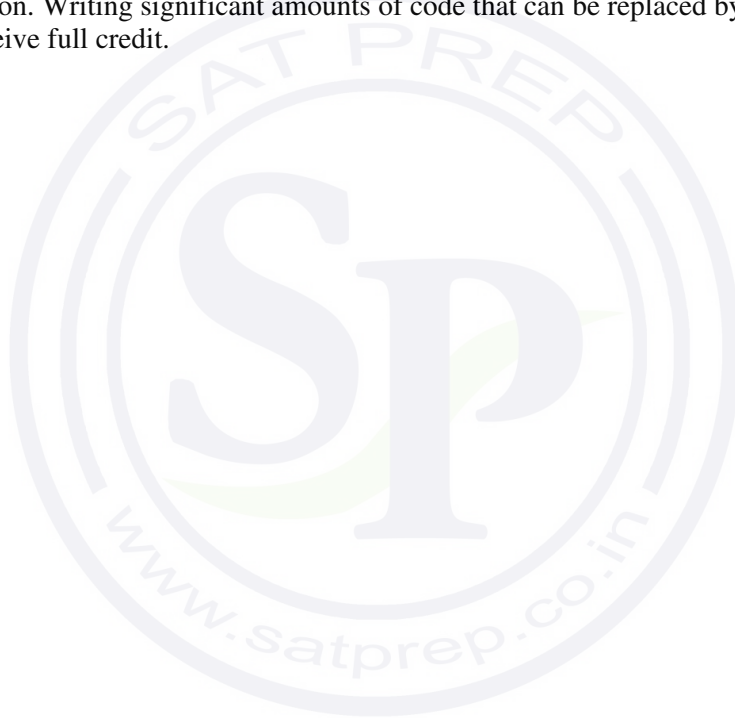
Number of questions—4

Percent of total score—50

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the interface and classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.



2017 AP[®] 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[®] 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[®] 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[®] 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[®] 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[®] 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[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

The following examples illustrate the behavior of the `MultiPractice` class. Each table shows a code segment and the output that would be produced as the code is executed.

Example 1

Code segment	Output produced
<pre>StudyPractice p1 = new MultiPractice(7, 3); System.out.println(p1.getProblem()); p1.nextProblem(); System.out.println(p1.getProblem()); p1.nextProblem(); System.out.println(p1.getProblem()); p1.nextProblem(); System.out.println(p1.getProblem());</pre>	<pre>7 TIMES 3 7 TIMES 4 7 TIMES 5 7 TIMES 6</pre>

Example 2

Code segment	Output produced
<pre>StudyPractice p2 = new MultiPractice(4, 12); p2.nextProblem(); System.out.println(p2.getProblem()); System.out.println(p2.getProblem()); p2.nextProblem(); p2.nextProblem(); System.out.println(p2.getProblem()); p2.nextProblem(); System.out.println(p2.getProblem());</pre>	<pre>4 TIMES 13 4 TIMES 13 4 TIMES 15 4 TIMES 16</pre>

Question 2 continues on page 10.

2017 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Interface information for this question

```
public interface StudyPractice
```

```
String getProblem()
```

```
void nextProblem()
```

Write the complete `MultiPractice` class. Your implementation must be consistent with the specifications and the given examples.



2017 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

3. This question involves analyzing and modifying a string. The following `Phrase` class maintains a phrase in an instance variable and has methods that access and make changes to the phrase. You will write two methods of the `Phrase` class.

```
public class Phrase
{
    private String currentPhrase;

    /** Constructs a new Phrase object. */
    public Phrase(String p)
    {   currentPhrase = p;   }

    /** Returns the index of the nth occurrence of str in the current phrase;
     *   returns -1 if the nth occurrence does not exist.
     *   Precondition: str.length() > 0 and n > 0
     *   Postcondition: the current phrase is not modified.
     */
    public int findNthOccurrence(String str, int n)
    {   /* implementation not shown */   }

    /** 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)
    {   /* to be implemented in part (a) */   }

    /** 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)
    {   /* to be implemented in part (b) */   }

    /** Returns a string containing the current phrase. */
    public String toString()
    {   return currentPhrase;   }
}
```

Part (a) begins on page 12.

2017 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the `Phrase` method `replaceNthOccurrence`, which will replace the `nth` occurrence of the string `str` with the string `repl`. If the `nth` occurrence does not exist, `currentPhrase` remains unchanged.

Several examples of the behavior of the method `replaceNthOccurrence` are shown below.

Code segments	Output produced
<pre>Phrase phrase1 = new Phrase("A cat ate late."); phrase1.replaceNthOccurrence("at", 1, "rane"); System.out.println(phrase1);</pre>	A crane ate late.
<pre>Phrase phrase2 = new Phrase("A cat ate late."); phrase2.replaceNthOccurrence("at", 6, "xx"); System.out.println(phrase2);</pre>	A cat ate late.
<pre>Phrase phrase3 = new Phrase("A cat ate late."); phrase3.replaceNthOccurrence("bat", 2, "xx"); System.out.println(phrase3);</pre>	A cat ate late.
<pre>Phrase phrase4 = new Phrase("aaaa"); phrase4.replaceNthOccurrence("aa", 1, "xx"); System.out.println(phrase4);</pre>	xxaa
<pre>Phrase phrase5 = new Phrase("aaaa"); phrase5.replaceNthOccurrence("aa", 2, "bbb"); System.out.println(phrase5);</pre>	abbba

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()
```


2017 AP[®] 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[®] 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[®] 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[®] 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`.

2017 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `findPosition` below.

```
/** Returns the position of num in intArr;  
 * returns null if no such element exists in intArr.  
 * Precondition: intArr contains at least one row.  
 */  
public static Position findPosition(int num, int[] [] intArr)
```



Part (b) begins on page 18.

2017 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write a static method `getSuccessorArray` that returns a 2D successor array of positions created from a given 2D integer array.

The *successor* of an integer value is the integer that is one greater than that value. For example, the successor of 8 is 9. A 2D *successor array* shows the position of the successor of each element in a given 2D integer array. The 2D successor array has the same dimensions as the given 2D integer array. Each element in the 2D successor array is the position (row, column) of the corresponding 2D integer array element's successor. The largest element in the 2D integer array does not have a successor in the 2D integer array, so its corresponding position in the 2D successor array is `null`.

The following diagram shows a 2D integer array and its corresponding 2D successor array. To illustrate the successor relationship, the values 8 and 9 in the 2D integer array are shaded. In the 2D successor array, the shaded element shows that the position of the successor of 8 is (0, 2) in the 2D integer array. The largest value in the 2D integer array is 16, so its corresponding element in the 2D successor array is `null`.

<u>2D Integer Array</u>					<u>2D Successor Array</u>				
	0	1	2	3		0	1	2	3
0	15	5	9	10	0	(1, 1)	(1, 3)	(0, 3)	(1, 2)
1	12	16	11	6	1	(2, 2)	null	(1, 0)	(2, 3)
2	14	8	13	7	2	(0, 0)	(0, 2)	(2, 0)	(2, 1)

Class information for this question

```
public class Position
```

```
public Position(int r, int c)
```

```
public class Successors
```

```
public static Position findPosition(int num, int[] [] intArr)
```

```
public static Position[] [] getSuccessorArray(int[] [] intArr)
```

2017 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Assume that `findPosition` works as specified, regardless of what you wrote in part (a). You must use `findPosition` appropriately to receive full credit.

Complete method `getSuccessorArray` below.

```
/** Returns a 2D successor array as described in part (b) constructed from intArr.  
 * Precondition: intArr contains at least one row and contains consecutive values.  
 * Each of these integers may be in any position in the 2D array.  
 */  
public static Position[] [] getSuccessorArray(int[] [] intArr)
```



STOP

END OF EXAM



AP[®] Computer Science A

2016 Free-Response Questions

© 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 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

COMPUTER SCIENCE A SECTION II

Time—1 hour and 30 minutes

Number of questions—4

Percent of total score—50

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

1. This question involves the implementation and extension of a `RandomStringChooser` class.

- (a) A `RandomStringChooser` object is constructed from an array of non-null `String` values. When the object is first constructed, all of the strings are considered available. The `RandomStringChooser` class has a `getNext` method, which has the following behavior. A call to `getNext` returns a randomly chosen string from the available strings in the object. Once a particular string has been returned from a call to `getNext`, it is no longer available to be returned from subsequent calls to `getNext`. If no strings are available to be returned, `getNext` returns `"NONE"`.

The following code segment shows an example of the behavior of `RandomStringChooser`.

```
String[] wordArray = {"wheels", "on", "the", "bus"};
RandomStringChooser sChooser = new RandomStringChooser(wordArray);
for (int k = 0; k < 6; k++)
{
    System.out.print(sChooser.getNext() + " ");
}
```

One possible output is shown below. Because `sChooser` has only four strings, the string `"NONE"` is printed twice.

```
bus the wheels on NONE NONE
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Write the entire `RandomStringChooser` class. Your implementation must include an appropriate constructor and any necessary methods. Any instance variables must be `private`. The code segment in the example above should have the indicated behavior (that is, it must compile and produce a result like the possible output shown). Neither the constructor nor any of the methods should alter the parameter passed to the constructor, but your implementation may copy the contents of the array.



Part (b) begins on page 4.

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) The following partially completed `RandomLetterChooser` class is a subclass of the `RandomStringChooser` class. You will write the constructor for the `RandomLetterChooser` class.

```
public class RandomLetterChooser extends RandomStringChooser
{
    /** Constructs a random letter chooser using the given string str.
     *   Precondition: str contains only letters.
     */
    public RandomLetterChooser(String str)
    { /* to be implemented in part (b) */ }

    /** Returns an array of single-letter strings.
     *   Each of these strings consists of a single letter from str. Element k
     *   of the returned array contains the single letter at position k of str.
     *   For example, getSingleLetters("cat") returns the
     *   array { "c", "a", "t" }.
     */
    public static String[] getSingleLetters(String str)
    { /* implementation not shown */ }
}
```

The following code segment shows an example of using `RandomLetterChooser`.

```
RandomLetterChooser letterChooser = new RandomLetterChooser("cat");
for (int k = 0; k < 4; k++)
{
    System.out.print(letterChooser.getNext());
}
```

The code segment will print the three letters in "cat" in one of the possible orders. Because there are only three letters in the original string, the code segment prints "NONE" the fourth time through the loop. One possible output is shown below.

actNONE

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Assume that the `RandomStringChooser` class that you wrote in part (a) has been implemented correctly and that `getSingleLetters` works as specified. You must use `getSingleLetters` appropriately to receive full credit.

Complete the `RandomLetterChooser` constructor below.

```
/** Constructs a random letter chooser using the given string str.  
 * Precondition: str contains only letters.  
 */  
public RandomLetterChooser(String str)
```



2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

2. This question involves two classes that are used to process log messages. A list of sample log messages is given below.

```
CLIENT3:security alert - repeated login failures
Webserver:disk offline
SERVER1:file not found
SERVER2:read error on disk DSK1
SERVER1:write error on disk DSK2
Webserver:error on /dev/disk
```

Log messages have the format *machineId:description*, where *machineId* identifies the computer and *description* describes the event being logged. Exactly one colon (":") appears in a log message. There are no blanks either immediately before or immediately after the colon.

The following `LogMessage` class is used to represent a log message.

```
public class LogMessage
{
    private String machineId;
    private String description;

    /** Precondition: message is a valid log message. */
    public LogMessage(String message)
    { /* to be implemented in part (a) */ }

    /** Returns true if the description in this log message properly contains keyword;
     *     false otherwise.
     */
    public boolean containsWord(String keyword)
    { /* to be implemented in part (b) */ }

    public String getMachineId()
    { return machineId; }

    public String getDescription()
    { return description; }

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

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the constructor for the `LogMessage` class. It must initialize the private data of the object so that `getMachineId` returns the *machineId* part of the message and `getDescription` returns the *description* part of the message.

Complete the `LogMessage` constructor below.

```
/** Precondition: message is a valid log message. */  
public LogMessage(String message)
```



Part (b) begins on page 8.

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `LogMessage` method `containsWord`, which returns `true` if the description in the log message *properly contains* a given keyword and returns `false` otherwise.

A description *properly contains* a keyword if all three of the following conditions are true.

- the keyword is a substring of the description;
- the keyword is either at the beginning of the description or it is immediately preceded by a space;
- the keyword is either at the end of the description or it is immediately followed by a space.

The following tables show several examples. The descriptions in the left table properly contain the keyword `"disk"`. The descriptions in the right table do not properly contain the keyword `"disk"`.

Descriptions that properly contain `"disk"`

<code>"disk"</code>
<code>"error on disk"</code>
<code>"error on /dev/disk disk"</code>
<code>"error on disk DSK1"</code>

Descriptions that do not properly contain `"disk"`

<code>"DISK"</code>
<code>"error on disk3"</code>
<code>"error on /dev/disk"</code>
<code>"diskette"</code>

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Assume that the `LogMessage` constructor works as specified, regardless of what you wrote in part (a).
Complete method `containsWord` below.

```
/** Returns true if the description in this log message properly contains keyword;  
 *      false otherwise.  
 */  
public boolean containsWord(String keyword)
```



Part (c) begins on page 10.

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (c) The `SystemLog` class represents a list of `LogMessage` objects and provides a method that removes and returns a list of all log messages (if any) that properly contain a given keyword. The messages in the returned list appear in the same order in which they originally appeared in the system log. If no message properly contains the keyword, an empty list is returned. The declaration of the `SystemLog` class is shown below.

```
public class SystemLog
{
    /** Contains all the entries in this system log.
     *  Guaranteed not to be null and to contain only non-null entries.
     */
    private List<LogMessage> messageList;

    /** Removes from the system log all entries whose descriptions properly contain keyword,
     *  and returns a list (possibly empty) containing the removed entries.
     *  Postcondition:
     *  - Entries in the returned list properly contain keyword and
     *    are in the order in which they appeared in the system log.
     *  - The remaining entries in the system log do not properly contain keyword and
     *    are in their original order.
     *  - The returned list is empty if no messages properly contain keyword.
     */
    public List<LogMessage> removeMessages(String keyword)
    { /* to be implemented in part (c) */ }

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

Write the `SystemLog` method `removeMessages`, which removes from the system log all entries whose descriptions properly contain `keyword` and returns a list of the removed entries in their original order. For example, assume that `theLog` is a `SystemLog` object initially containing six `LogMessage` objects representing the following list of log messages.

```
CLIENT3:security alert - repeated login failures
Webserver:disk offline
SERVER1:file not found
SERVER2:read error on disk DSK1
SERVER1:write error on disk DSK2
Webserver:error on /dev/disk
```

The call `theLog.removeMessages("disk")` would return a list containing the `LogMessage` objects representing the following log messages.

```
Webserver:disk offline
SERVER2:read error on disk DSK1
SERVER1:write error on disk DSK2
```

After the call, `theLog` would contain the following log messages.

```
CLIENT3:security alert - repeated login failures
SERVER1:file not found
Webserver:error on /dev/disk
```

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Assume that the `LogMessage` class works as specified, regardless of what you wrote in parts (a) and (b). You must use `containsWord` appropriately to receive full credit.

Complete method `removeMessages` below.

```
/** Removes from the system log all entries whose descriptions properly contain keyword,
 * and returns a list (possibly empty) containing the removed entries.
 * Postcondition:
 *   - Entries in the returned list properly contain keyword and
 *     are in the order in which they appeared in the system log.
 *   - The remaining entries in the system log do not properly contain keyword and
 *     are in their original order.
 *   - The returned list is empty if no messages properly contain keyword.
 */
public List<LogMessage> removeMessages(String keyword)
```



2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

3. A crossword puzzle grid is a two-dimensional rectangular array of black and white squares. Some of the white squares are labeled with a positive number according to the *crossword labeling rule*.

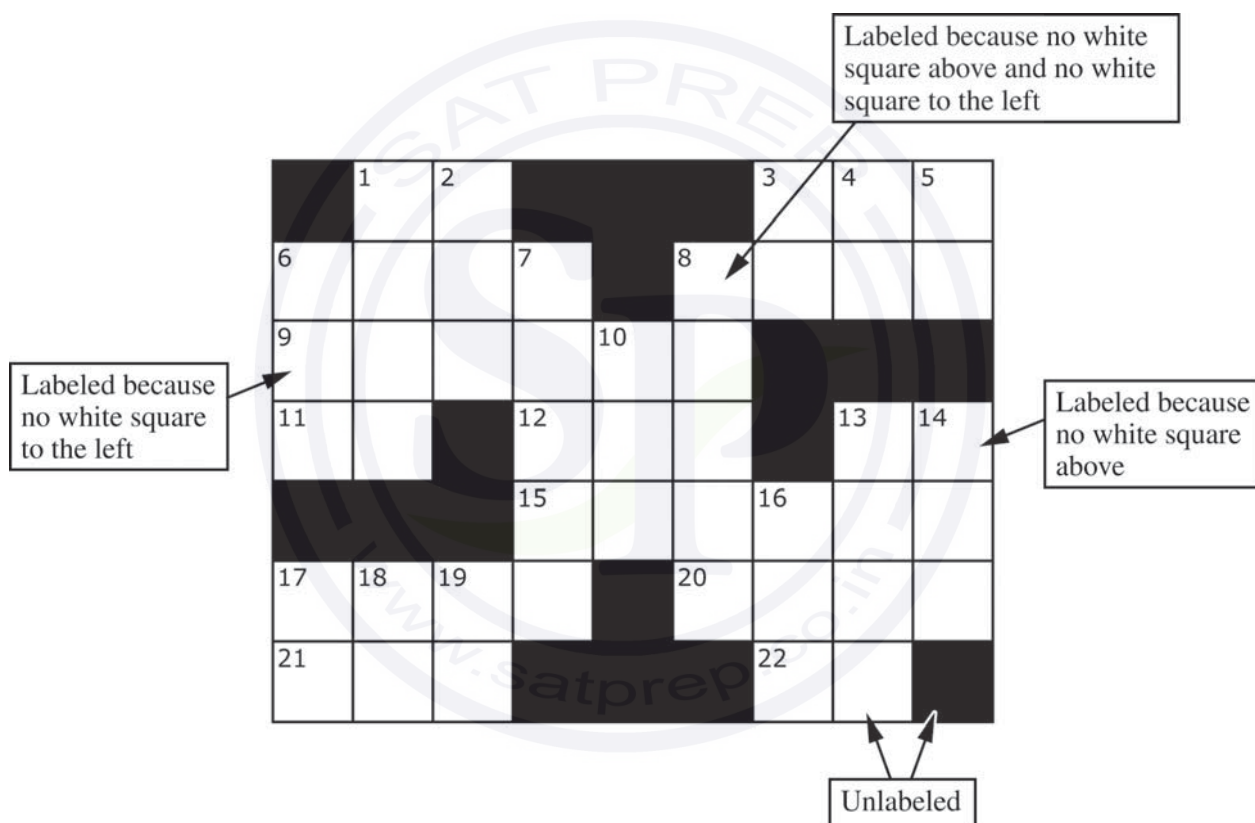
The crossword labeling rule identifies squares to be labeled with a positive number as follows.

A square is labeled with a positive number if and only if

- the square is white and
- the square does not have a white square immediately above it, or it does not have a white square immediately to its left, or both.

The squares identified by these criteria are labeled with consecutive numbers in row-major order, starting at 1.

The following diagram shows a crossword puzzle grid and the labeling of the squares according to the crossword labeling rule.



2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

This question uses two classes, a `Square` class that represents an individual square in the puzzle and a `Crossword` class that represents a crossword puzzle grid. A partial declaration of the `Square` class is shown below.

```
public class Square
{
    /** Constructs one square of a crossword puzzle grid.
     * Postcondition:
     *     - The square is black if and only if isBlack is true.
     *     - The square has number num.
     */
    public Square(boolean isBlack, int num)
    { /* implementation not shown */ }

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

A partial declaration of the `Crossword` class is shown below. You will implement one method and the constructor in the `Crossword` class.

```
public class Crossword
{
    /** Each element is a Square object with a color (black or white) and a number.
     * puzzle[r][c] represents the square in row r, column c.
     * There is at least one row in the puzzle.
     */
    private Square[][] puzzle;

    /** Constructs a crossword puzzle grid.
     * Precondition: There is at least one row in blackSquares.
     * Postcondition:
     *     - The crossword puzzle grid has the same dimensions as blackSquares.
     *     - The Square object at row r, column c in the crossword puzzle grid is black
     *       if and only if blackSquares[r][c] is true.
     *     - The squares in the puzzle are labeled according to the crossword labeling rule.
     */
    public Crossword(boolean[][] blackSquares)
    { /* to be implemented in part (b) */ }

    /** Returns true if the square at row r, column c should be labeled with a positive number;
     *     false otherwise.
     * The square at row r, column c is black if and only if blackSquares[r][c] is true.
     * Precondition: r and c are valid indexes in blackSquares.
     */
    private boolean toBeLabeled(int r, int c, boolean[][] blackSquares)
    { /* to be implemented in part (a) */ }

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

Part (a) begins on page 14.

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the `Crossword` method `toBeLabeled`. The method returns `true` if the square indexed by row `r`, column `c` in a crossword puzzle grid should be labeled with a positive number according to the crossword labeling rule; otherwise it returns `false`. The parameter `blackSquares` indicates which squares in the crossword puzzle grid are black.

Class information for this question

```
public class Square
```

```
public Square(boolean isBlack, int num)
```

```
public class Crossword
```

```
private Square[][] puzzle
```

```
public Crossword(boolean[][] blackSquares)
```

```
private boolean toBeLabeled(int r, int c, boolean[][] blackSquares)
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `toBeLabeled` below.

```
/** Returns true if the square at row r, column c should be labeled with a positive number;  
 *     false otherwise.  
 *     The square at row r, column c is black if and only if blackSquares[r][c] is true.  
 *     Precondition: r and c are valid indexes in blackSquares.  
 */  
private boolean toBeLabeled(int r, int c, boolean[][] blackSquares)
```



Part (b) begins on page 16.

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `Crossword` constructor. The constructor should initialize the crossword puzzle grid to have the same dimensions as the parameter `blackSquares`. Each element of the puzzle grid should be initialized with a reference to a `Square` object with the appropriate color and number. The number is positive if the square is labeled and 0 if the square is not labeled.

Class information for this question

```
public class Square
```

```
public Square(boolean isBlack, int num)
```

```
public class Crossword
```

```
private Square[][] puzzle
```

```
public Crossword(boolean[][] blackSquares)
```

```
private boolean toBeLabeled(int r, int c, boolean[][] blackSquares)
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Assume that `toBeLabeled` works as specified, regardless of what you wrote in part (a). You must use `toBeLabeled` appropriately to receive full credit.

Complete the `Crossword` constructor below.

```
/** Constructs a crossword puzzle grid.
 * Precondition: There is at least one row in blackSquares.
 * Postcondition:
 *   - The crossword puzzle grid has the same dimensions as blackSquares.
 *   - The Square object at row r, column c in the crossword puzzle grid is black
 *     if and only if blackSquares[r][c] is true.
 *   - The squares in the puzzle are labeled according to the crossword labeling rule.
 */
public Crossword(boolean[][] blackSquares)
```



2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

4. This question involves the process of taking a list of words, called `wordList`, and producing a formatted string of a specified length. The list `wordList` contains at least two words, consisting of letters only.

When the formatted string is constructed, spaces are placed in the gaps between words so that as many spaces as possible are evenly distributed to each gap. The equal number of spaces inserted into each gap is referred to as the *basic gap width*. Any *leftover spaces* are inserted one at a time into the gaps from left to right until there are no more leftover spaces.

The following three examples illustrate these concepts. In each example, the list of words is to be placed into a formatted string of length 20.

Example 1: wordList: ["AP", "COMP", "SCI", "ROCKS"]

Total number of letters in words: 14

Number of gaps between words: 3

Basic gap width: 2

Leftover spaces: 0

Formatted string:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	P			C	O	M	P			S	C	I			R	O	C	K	S

Example 2: wordList: ["GREEN", "EGGS", "AND", "HAM"]

Total number of letters in words: 15

Number of gaps between words: 3

Basic gap width: 1

Leftover spaces: 2

The leftover spaces are inserted one at a time between the words from left to right until there are no more leftover spaces. In this example, the first two gaps get an extra space.

Formatted string:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
G	R	E	E	N			E	G	G	S			A	N	D			H	A	M

Example 3: wordList: ["BEACH", "BALL"]

Total number of letters in words: 9

Number of gaps between words: 1

Basic gap width: 11

Leftover spaces: 0

Formatted string:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
B	E	A	C	H												B	A	L	L

You will implement three `static` methods in a class named `StringFormatter` that is not shown.

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the `StringFormatter` method `totalLetters`, which returns the total number of letters in the words in its parameter `wordList`. For example, if the variable `List<String> words` is `["A", "frog", "is"]`, then the call `StringFormatter.totalLetters(words)` returns 7. You may assume that all words in `wordList` consist of one or more letters.

Complete method `totalLetters` below.

```
/** Returns the total number of letters in wordList.  
 * Precondition: wordList contains at least two words, consisting of letters only.  
 */  
public static int totalLetters(List<String> wordList)
```



Part (b) begins on page 20.

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `StringFormatter` method `basicGapWidth`, which returns the basic gap width as defined earlier.

Class information for this question

```
public class StringFormatter  
  
public static int totalLetters(List<String> wordList)  
public static int basicGapWidth(List<String> wordList,  
                                int formattedLen)  
public static int leftoverSpaces(List<String> wordList,  
                                int formattedLen)  
public static String format(List<String> wordList, int formattedLen)
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.



2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Assume that `totalLetters` works as specified regardless of what you wrote in part (a). You must use `totalLetters` appropriately to receive full credit.

Complete method `basicGapWidth` below.

```
/** Returns the basic gap width when wordList is used to produce
 *   a formatted string of formattedLen characters.
 *   Precondition: wordList contains at least two words, consisting of letters only.
 *                   formattedLen is large enough for all the words and gaps.
 */
public static int basicGapWidth(List<String> wordList,
                                int formattedLen)
```



Part (c) begins on page 22.

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (c) Write the `StringFormatter` method `format`, which returns the formatted string as defined earlier. The `StringFormatter` class also contains a method called `leftoverSpaces`, which has already been implemented. This method returns the number of leftover spaces as defined earlier and is shown below.

```
/** Returns the number of leftover spaces when wordList is used to produce
 * a formatted string of formattedLen characters.
 * Precondition: wordList contains at least two words, consisting of letters only.
 * formattedLen is large enough for all the words and gaps.
 */
public static int leftoverSpaces(List<String> wordList,
                                int formattedLen)
{ /* implementation not shown */ }
```

Class information for this question

```
public class StringFormatter

public static int totalLetters(List<String> wordList)
public static int basicGapWidth(List<String> wordList,
                                int formattedLen)
public static int leftoverSpaces(List<String> wordList,
                                int formattedLen)
public static String format(List<String> wordList, int formattedLen)
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2016 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Assume that `basicGapWidth` works as specified, regardless of what you wrote in part (b). You must use `basicGapWidth` and `leftoverSpaces` appropriately to receive full credit.

Complete method `format` below.

```
/** Returns a formatted string consisting of the words in wordList separated by spaces.
 * Precondition: The wordList contains at least two words, consisting of letters only.
 * formattedLen is large enough for all the words and gaps.
 * Postcondition: All words in wordList appear in the formatted string.
 * - The words appear in the same order as in wordList.
 * - The number of spaces between words is determined by basicGapWidth and the
 *   distribution of leftoverSpaces from left to right, as described in the question.
 */
public static String format(List<String> wordList, int formattedLen)
```



STOP

END OF EXAM



AP[®] Computer Science A

2015 Free-Response Questions

© 2015 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.

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

COMPUTER SCIENCE A SECTION II

Time—1 hour and 45 minutes

Number of questions—4

Percent of total score—50

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
 - Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
 - In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.
1. This question involves reasoning about one-dimensional and two-dimensional arrays of integers. You will write three static methods, all of which are in a single enclosing class, named `DiverseArray` (not shown). The first method returns the sum of the values of a one-dimensional array; the second method returns an array that represents the sums of the rows of a two-dimensional array; and the third method analyzes row sums.
- (a) Write a static method `arraySum` that calculates and returns the sum of the entries in a specified one-dimensional array. The following example shows an array `arr1` and the value returned by a call to `arraySum`.

<u>arr1</u>					Value returned by <u>arraySum(arr1)</u>
0	1	2	3	4	
1	3	2	7	3	16

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `arraySum` below.

```
/** Returns the sum of the entries in the one-dimensional array arr.  
 */  
public static int arraySum(int[] arr)
```



Part (b) begins on page 4.

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write a static method `rowSums` that calculates the sums of each of the rows in a given two-dimensional array and returns these sums in a one-dimensional array. The method has one parameter, a two-dimensional array `arr2D` of `int` values. The array is in row-major order: `arr2D[r][c]` is the entry at row `r` and column `c`. The method returns a one-dimensional array with one entry for each row of `arr2D` such that each entry is the sum of the corresponding row in `arr2D`. As a reminder, each row of a two-dimensional array is a one-dimensional array.

For example, if `mat1` is the array represented by the following table, the call `rowSums(mat1)` returns the array `{16, 32, 28, 20}`.

	<u>mat1</u>				
	0	1	2	3	4
0	1	3	2	7	3
1	10	10	4	6	2
2	5	3	5	9	6
3	7	6	4	2	1

Methods written in this question

```
public static int arraySum(int[] arr)
public static int[] rowSums(int[][] arr2D)
public static boolean isDiverse(int[][] arr2D)
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Assume that `arraySum` works as specified, regardless of what you wrote in part (a). You must use `arraySum` appropriately to receive full credit.

Complete method `rowSums` below.

```
/** Returns a one-dimensional array in which the entry at index k is the sum of
 * the entries of row k of the two-dimensional array arr2D.
 */
public static int[] rowSums(int[][] arr2D)
```



Part (c) begins on page 6.

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (c) A two-dimensional array is *diverse* if no two of its rows have entries that sum to the same value. In the following examples, the array `mat1` is diverse because each row sum is different, but the array `mat2` is not diverse because the first and last rows have the same sum.

<u>mat1</u>						
	0	1	2	3	4	Row sums
0	1	3	2	7	3	16
1	10	10	4	6	2	32
2	5	3	5	9	6	28
3	7	6	4	2	1	20

<u>mat2</u>						
	0	1	2	3	4	Row sums
0	1	1	5	3	4	14
1	12	7	6	1	9	35
2	8	11	10	2	5	36
3	3	2	3	0	6	14

Write a static method `isDiverse` that determines whether or not a given two-dimensional array is diverse. The method has one parameter: a two-dimensional array `arr2D` of `int` values. The method should return `true` if all the row sums in the given array are unique; otherwise, it should return `false`. In the arrays shown above, the call `isDiverse(mat1)` returns `true` and the call `isDiverse(mat2)` returns `false`.

Methods written in this question

```
public static int arraySum(int[] arr)
public static int[] rowSums(int[][] arr2D)
public static boolean isDiverse(int[][] arr2D)
```

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Assume that `arraySum` and `rowSums` work as specified, regardless of what you wrote in parts (a) and (b). You must use `rowSums` appropriately to receive full credit.

Complete method `isDiverse` below.

```
/** Returns true if all rows in arr2D have different row sums;  
 *     false otherwise.  
 */  
public static boolean isDiverse(int[][] arr2D)
```



2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

2. Consider a guessing game in which a player tries to guess a hidden word. The hidden word contains only capital letters and has a length known to the player. A guess contains only capital letters and has the same length as the hidden word.

After a guess is made, the player is given a hint that is based on a comparison between the hidden word and the guess. Each position in the hint contains a character that corresponds to the letter in the same position in the guess. The following rules determine the characters that appear in the hint.

If the letter in the guess is ...	the corresponding character in the hint is
also in the same position in the hidden word,	the matching letter
also in the hidden word, but in a different position,	" + "
not in the hidden word,	" * "

The `HiddenWord` class will be used to represent the hidden word in the game. The hidden word is passed to the constructor. The class contains a method, `getHint`, that takes a guess and produces a hint.

For example, suppose the variable `puzzle` is declared as follows.

```
HiddenWord puzzle = new HiddenWord("HARPS");
```

The following table shows several guesses and the hints that would be produced.

Call to <code>getHint</code>	String returned
<code>puzzle.getHint("AAAAA")</code>	<code>" +A+++ "</code>
<code>puzzle.getHint("HELLO")</code>	<code>"H**** "</code>
<code>puzzle.getHint("HEART")</code>	<code>"H*++* "</code>
<code>puzzle.getHint("HARMS")</code>	<code>"HAR*S "</code>
<code>puzzle.getHint("HARPS")</code>	<code>"HARPS "</code>

Write the complete `HiddenWord` class, including any necessary instance variables, its constructor, and the method, `getHint`, described above. You may assume that the length of the guess is the same as the length of the hidden word.

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

3. A two-dimensional array of integers in which most elements are zero is called a *sparse array*. Because most elements have a value of zero, memory can be saved by storing only the non-zero values along with their row and column indexes. The following complete `SparseArrayEntry` class is used to represent non-zero elements in a sparse array. A `SparseArrayEntry` object cannot be modified after it has been constructed.

```
public class SparseArrayEntry
{
    /** The row index and column index for this entry in the sparse array */
    private int row;
    private int col;

    /** The value of this entry in the sparse array */
    private int value;

    /** Constructs a SparseArrayEntry object that represents a sparse array element
     * with row index r and column index c, containing value v.
     */
    public SparseArrayEntry(int r, int c, int v)
    {
        row = r;
        col = c;
        value = v;
    }

    /** Returns the row index of this sparse array element. */
    public int getRow()
    { return row; }

    /** Returns the column index of this sparse array element. */
    public int getCol()
    { return col; }

    /** Returns the value of this sparse array element. */
    public int getValue()
    { return value; }
}
```

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

The `SparseArray` class represents a sparse array. It contains a list of `SparseArrayEntry` objects, each of which represents one of the non-zero elements in the array. The entries representing the non-zero elements are stored in the list in no particular order. Each non-zero element is represented by exactly one entry in the list.

```
public class SparseArray
{
    /** The number of rows and columns in the sparse array. */
    private int numRows;
    private int numCols;

    /** The list of entries representing the non-zero elements of the sparse array. Entries are stored in the
     *  list in no particular order. Each non-zero element is represented by exactly one entry in the list.
     */
    private List<SparseArrayEntry> entries;

    /** Constructs an empty SparseArray. */
    public SparseArray()
    { entries = new ArrayList<SparseArrayEntry>(); }

    /** Returns the number of rows in the sparse array. */
    public int getNumRows()
    { return numRows; }

    /** Returns the number of columns in the sparse array. */
    public int getNumCols()
    { return numCols; }

    /** Returns the value of the element at row index row and column index col in the sparse array.
     *  Precondition:  $0 \leq \text{row} < \text{getNumRows}()$ 
     *                   $0 \leq \text{col} < \text{getNumCols}()$ 
     */
    public int getValueAt(int row, int col)
    { /* to be implemented in part (a) */ }

    /** Removes the column col from the sparse array.
     *  Precondition:  $0 \leq \text{col} < \text{getNumCols}()$ 
     */
    public void removeColumn(int col)
    { /* to be implemented in part (b) */ }

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


2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

The following table shows an example of a two-dimensional sparse array. Empty cells in the table indicate zero values.

	0	1	2	3	4
0					
1		5			4
2	1				
3		-9			
4					
5					

The sample array can be represented by a `SparseArray` object, `sparse`, with the following instance variable values. The items in `entries` are in no particular order; one possible ordering is shown below.

`numRows: 6`

`numCols: 5`

`entries:`

<code>row: 1</code> <code>col: 4</code> <code>value: 4</code>	<code>row: 2</code> <code>col: 0</code> <code>value: 1</code>	<code>row: 3</code> <code>col: 1</code> <code>value: -9</code>	<code>row: 1</code> <code>col: 1</code> <code>value: 5</code>
---	---	--	---

- (a) Write the `SparseArray` method `getValueAt`. The method returns the value of the sparse array element at a given row and column in the sparse array. If the list `entries` contains an entry with the specified row and column, the value associated with the entry is returned. If there is no entry in `entries` corresponding to the specified row and column, 0 is returned.

In the example above, the call `sparse.getValueAt(3, 1)` would return -9, and `sparse.getValueAt(3, 3)` would return 0.

WRITE YOUR SOLUTION ON THE NEXT PAGE.

Part (a) continues on page 12.

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `getValueAt` below.

```
/** Returns the value of the element at row index row and column index col in the sparse array.  
 * Precondition:  $0 \leq \text{row} < \text{getNumRows}()$   
 *  $0 \leq \text{col} < \text{getNumCols}()$   
 */  
public int getValueAt(int row, int col)
```



Part (b) begins on page 13.

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

(b) Write the `SparseArray` method `removeColumn`. After removing a specified column from a sparse array:

- All entries in the list `entries` with column indexes matching `col` are removed from the list.
- All entries in the list `entries` with column indexes greater than `col` are replaced by entries with column indexes that are decremented by one (moved one column to the left).
- The number of columns in the sparse array is adjusted to reflect the column removed.

The sample object `sparse` from the beginning of the question is repeated for your convenience.

	0	1	2	3	4
0					
1		5			4
2	1				
3		-9			
4					
5					

The shaded entries in `entries`, below, correspond to the shaded column above.

`numRows: 6`

`numCols: 5`

<code>entries:</code>	<div>row: 1 col: 4 value: 4</div>	<div>row: 2 col: 0 value: 1</div>	<div>row: 3 col: 1 value: -9</div>	<div>row: 1 col: 1 value: 5</div>
-----------------------	---	---	--	---

When `sparse` has the state shown above, the call `sparse.removeColumn(1)` could result in `sparse` having the following values in its instance variables (since `entries` is in no particular order, it would be equally valid to reverse the order of its two items). The shaded areas below show the changes.

`numRows: 6`

`numCols: 4`

<code>entries:</code>	<div>row: 1 col: 3 value: 4</div>	<div>row: 2 col: 0 value: 1</div>
-----------------------	---	---

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Class information repeated from the beginning of the question

```
public class SparseArrayEntry
```

```
public SparseArrayEntry(int r, int c, int v)
public int getRow()
public int getCol()
public int getValue()
```

```
public class SparseArray
```

```
private int numRows
private int numCols
private List<SparseArrayEntry> entries
public int getNumRows()
public int getNumCols()
public int getValueAt(int row, int col)
public void removeColumn(int col)
```



2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `removeColumn` below.

```
/** Removes the column col from the sparse array.  
 * Precondition:  $0 \leq \text{col} < \text{getNumCols}()$   
 */  
public void removeColumn(int col)
```



2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

4. This question involves the design of an interface, writing a class that implements the interface, and writing a method that uses the interface.

- (a) A *number group* represents a group of integers defined in some way. It could be empty, or it could contain one or more integers.

Write an interface named `NumberGroup` that represents a group of integers. The interface should have a single `contains` method that determines if a given integer is in the group. For example, if `group1` is of type `NumberGroup`, and it contains only the two numbers `-5` and `3`, then `group1.contains(-5)` would return `true`, and `group1.contains(2)` would return `false`.

Write the complete `NumberGroup` interface. It must have exactly one method.



Part (b) begins on page 17.

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) A *range* represents a number group that contains all (and only) the integers between a minimum value and a maximum value, inclusive.

Write the `Range` class, which is a `NumberGroup`. The `Range` class represents the group of `int` values that range from a given minimum value up through a given maximum value, inclusive. For example, the declaration

```
NumberGroup range1 = new Range(-3, 2);
```

represents the group of integer values -3, -2, -1, 0, 1, 2.

Write the complete `Range` class. Include all necessary instance variables and methods as well as a constructor that takes two `int` parameters. The first parameter represents the minimum value, and the second parameter represents the maximum value of the range. You may assume that the minimum is less than or equal to the maximum.



Part (c) begins on page 18.

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (c) The `MultipleGroups` class (not shown) represents a collection of `NumberGroup` objects and is a `NumberGroup`. The `MultipleGroups` class stores the number groups in the instance variable `groupList` (shown below), which is initialized in the constructor.

```
private List<NumberGroup> groupList;
```

Write the `MultipleGroups` method `contains`. The method takes an integer and returns `true` if and only if the integer is contained in one or more of the number groups in `groupList`.

For example, suppose `multiple1` has been declared as an instance of `MultipleGroups` and consists of the three ranges created by the calls `new Range(5, 8)`, `new Range(10, 12)`, and `new Range(1, 6)`. The following table shows the results of several calls to `contains`.

Call	Result
<code>multiple1.contains(2)</code>	<code>true</code>
<code>multiple1.contains(9)</code>	<code>false</code>
<code>multiple1.contains(6)</code>	<code>true</code>

2015 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `contains` below.

```
/** Returns true if at least one of the number groups in this multiple group contains num;  
 *      false otherwise.  
 */  
public boolean contains(int num)
```



STOP

END OF EXAM



AP[®] Computer Science A

2014 Free-Response Questions

© 2014 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.

2014 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS**COMPUTER SCIENCE A
SECTION II****Time—1 hour and 45 minutes****Number of questions—4****Percent of total score—50**

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the appendices have been imported where appropriate.
 - Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
 - In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.
1. This question involves reasoning about strings made up of uppercase letters. You will implement two related methods that appear in the same class (not shown). The first method takes a single string parameter and returns a scrambled version of that string. The second method takes a list of strings and modifies the list by scrambling each entry in the list. Any entry that cannot be scrambled is removed from the list.
- (a) Write the method `scrambleWord`, which takes a given word and returns a string that contains a scrambled version of the word according to the following rules.
- The scrambling process begins at the first letter of the word and continues from left to right.
 - If two consecutive letters consist of an "A" followed by a letter that is not an "A", then the two letters are swapped in the resulting string.
 - Once the letters in two adjacent positions have been swapped, neither of those two positions can be involved in a future swap.

The following table shows several examples of words and their scrambled versions.

word	Result returned by <code>scrambleWord(word)</code>
"TAN"	"TNA"
"ABRACADABRA"	"BARCADABARA"
"WHOA"	"WHOA"
"AARDVARK"	"ARADVRAK"
"EGGS"	"EGGS"
"A"	"A"
""	""

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2014 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `scrambleWord` below.

```
/** Scrambles a given word.
 * @param word the word to be scrambled
 * @return the scrambled word (possibly equal to word)
 * Precondition: word is either an empty string or contains only uppercase letters.
 * Postcondition: the string returned was created from word as follows:
 *   - the word was scrambled, beginning at the first letter and continuing from left to right
 *   - two consecutive letters consisting of "A" followed by a letter that was not "A" were swapped
 *   - letters were swapped at most once
 */
public static String scrambleWord(String word)
```



Part (b) begins on page 4.

2014 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the method `scrambleOrRemove`, which replaces each word in the parameter `wordList` with its scrambled version and removes any words that are unchanged after scrambling. The relative ordering of the entries in `wordList` remains the same as before the call to `scrambleOrRemove`.

The following example shows how the contents of `wordList` would be modified as a result of calling `scrambleOrRemove`.

Before the call to `scrambleOrRemove`:

	0	1	2	3	4
wordList	"TAN"	"ABRACADABRA"	"WHOA"	"APPLE"	"EGGS"

After the call to `scrambleOrRemove`:

	0	1	2
wordList	"TNA"	"BARCADABARA"	"PAPLE"

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2014 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Assume that `scrambleWord` is in the same class as `scrambleOrRemove` and works as specified, regardless of what you wrote in part (a).

Complete method `scrambleOrRemove` below.

```
/** Modifies wordList by replacing each word with its scrambled
 * version, removing any words that are unchanged as a result of scrambling.
 * @param wordList the list of words
 * Precondition: wordList contains only non-null objects
 * Postcondition:
 *   - all words unchanged by scrambling have been removed from wordList
 *   - each of the remaining words has been replaced by its scrambled version
 *   - the relative ordering of the entries in wordList is the same as it was
 *     before the method was called
 */
public static void scrambleOrRemove(List<String> wordList)
```



2014 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

2. This question involves reasoning about the GridWorld case study. Reference materials are provided in the appendixes.

A `Director` is a type of `Rock` that has the following characteristics.

- A `Director` has an initial color of `Color.RED` and alternates between `Color.RED` and `Color.GREEN` each time it acts.
- If the color of a `Director` is `Color.GREEN` when it begins to act, it will cause any `Actor` objects in its neighboring cells to turn 90 degrees to their right.

Write the complete `Director` class, including the zero-parameter constructor and any necessary instance variables and methods. Assume that the `Color` class has been imported.



2014 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

3. A student in a school is represented by the following class.

```
public class Student
{
    /** Returns the name of this Student. */
    public String getName()
    { /* implementation not shown */ }

    /** Returns the number of times this Student has missed class. */
    public int getAbsenceCount()
    { /* implementation not shown */ }

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

The class `SeatingChart`, shown below, uses a two-dimensional array to represent the seating arrangement of students in a classroom. The seats in the classroom are in a rectangular arrangement of rows and columns.

```
public class SeatingChart
{
    /** seats[r][c] represents the Student in row r and column c in the classroom. */
    private Student[][] seats;

    /** Creates a seating chart with the given number of rows and columns from the students in
     * studentList. Empty seats in the seating chart are represented by null.
     * @param rows the number of rows of seats in the classroom
     * @param cols the number of columns of seats in the classroom
     * Precondition: rows > 0; cols > 0;
     *                  rows * cols >= studentList.size()
     * Postcondition:
     *   - Students appear in the seating chart in the same order as they appear
     *     in studentList, starting at seats[0][0].
     *   - seats is filled column by column from studentList, followed by any
     *     empty seats (represented by null).
     *   - studentList is unchanged.
     */
    public SeatingChart(List<Student> studentList,
                       int rows, int cols)
    { /* to be implemented in part (a) */ }

    /** Removes students who have more than a given number of absences from the
     * seating chart, replacing those entries in the seating chart with null
     * and returns the number of students removed.
     * @param allowedAbsences an integer >= 0
     * @return number of students removed from seats
     * Postcondition:
     *   - All students with allowedAbsences or fewer are in their original positions in seats.
     *   - No student in seats has more than allowedAbsences absences.
     *   - Entries without students contain null.
     */
    public int removeAbsentStudents(int allowedAbsences)
    { /* to be implemented in part (b) */ }

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


2014 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the constructor for the `SeatingChart` class. The constructor initializes the `seats` instance variable to a two-dimensional array with the given number of rows and columns. The students in `studentList` are copied into the seating chart in the order in which they appear in `studentList`. The students are assigned to consecutive locations in the array `seats`, starting at `seats[0][0]` and filling the array column by column. Empty seats in the seating chart are represented by `null`.

For example, suppose a variable `List<Student> roster` contains references to `Student` objects in the following order.

"Karen" 3	"Liz" 1	"Paul" 4	"Lester" 1	"Henry" 5	"Renee" 9	"Glen" 2	"Fran" 6	"David" 1	"Danny" 3
--------------	------------	-------------	---------------	--------------	--------------	-------------	-------------	--------------	--------------

A `SeatingChart` object created with the call `new SeatingChart(roster, 3, 4)` would have `seats` initialized with the following values.

	0	1	2	3
0	"Karen" 3	"Lester" 1	"Glen" 2	"Danny" 3
1	"Liz" 1	"Henry" 5	"Fran" 6	null
2	"Paul" 4	"Renee" 9	"David" 1	null

WRITE YOUR SOLUTION ON THE NEXT PAGE.

Part (a) continues on page 9.

2014 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete the `SeatingChart` constructor below.

```
/** Creates a seating chart with the given number of rows and columns from the students in
 *  studentList. Empty seats in the seating chart are represented by null.
 *  @param rows the number of rows of seats in the classroom
 *  @param cols the number of columns of seats in the classroom
 *  Precondition: rows > 0; cols > 0;
 *                  rows * cols >= studentList.size()
 *  Postcondition:
 *      - Students appear in the seating chart in the same order as they appear
 *        in studentList, starting at seats[0][0].
 *      - seats is filled column by column from studentList, followed by any
 *        empty seats (represented by null).
 *      - studentList is unchanged.
 */
public SeatingChart(List<Student> studentList,
                   int rows, int cols)
```

Part (b) begins on page 10.

2014 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `removeAbsentStudents` method, which removes students who have more than a given number of absences from the seating chart and returns the number of students that were removed. When a student is removed from the seating chart, a `null` is placed in the entry for that student in the array `seats`. For example, suppose the variable `SeatingChart introCS` has been created such that the array `seats` contains the following entries showing both students and their number of absences.

	0	1	2	3
0	"Karen" 3	"Lester" 1	"Glen" 2	"Danny" 3
1	"Liz" 1	"Henry" 5	"Fran" 6	<code>null</code>
2	"Paul" 4	"Renee" 9	"David" 1	<code>null</code>

After the call `introCS.removeAbsentStudents(4)` has executed, the array `seats` would contain the following values and the method would return the value 3.

	0	1	2	3
0	"Karen" 3	"Lester" 1	"Glen" 2	"Danny" 3
1	"Liz" 1	<code>null</code>	<code>null</code>	<code>null</code>
2	"Paul" 4	<code>null</code>	"David" 1	<code>null</code>

Class information repeated from the beginning of the question:

```
public class Student
{
    public String getName()
    public int getAbsenceCount()
}

public class SeatingChart
{
    private Student[][] seats
    public SeatingChart(List<Student> studentList,
                       int rows, int cols)
    public int removeAbsentStudents(int allowedAbsences)
}
```

2014 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `removeAbsentStudents` below.

```
/** Removes students who have more than a given number of absences from the
 * seating chart, replacing those entries in the seating chart with null
 * and returns the number of students removed.
 * @param allowedAbsences an integer  $\geq 0$ 
 * @return number of students removed from seats
 * Postcondition:
 *   - All students with allowedAbsences or fewer are in their original positions in seats.
 *   - No student in seats has more than allowedAbsences absences.
 *   - Entries without students contain null.
 */
public int removeAbsentStudents(int allowedAbsences)
```



2014 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

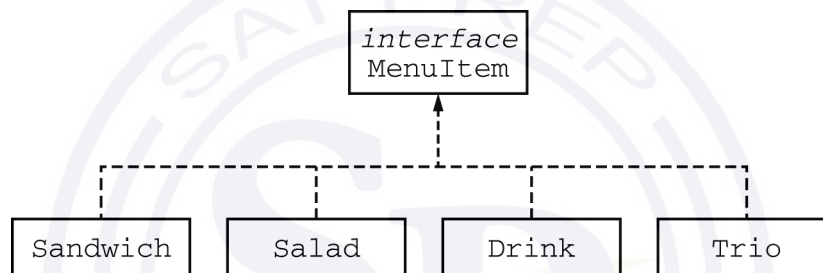
4. The menu at a lunch counter includes a variety of sandwiches, salads, and drinks. The menu also allows a customer to create a "trio," which consists of three menu items: a sandwich, a salad, and a drink. The price of the trio is the sum of the two highest-priced menu items in the trio; one item with the lowest price is free.

Each menu item has a name and a price. The four types of menu items are represented by the four classes Sandwich, Salad, Drink, and Trio. All four classes implement the following MenuItem interface.

```
public interface MenuItem
{
    /** @return the name of the menu item */
    String getName();

    /** @return the price of the menu item */
    double getPrice();
}
```

The following diagram shows the relationship between the MenuItem interface and the Sandwich, Salad, Drink, and Trio classes.



For example, assume that the menu includes the following items. The objects listed under each heading are instances of the class indicated by the heading.

Sandwich	Salad	Drink
"Cheeseburger" 2.75	"Spinach Salad" 1.25	"Orange Soda" 1.25
"Club Sandwich" 2.75	"Coleslaw" 1.25	"Cappuccino" 3.50

Question 4 continues on page 13

2014 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

The menu allows customers to create `Trio` menu items, each of which includes a sandwich, a salad, and a drink. The name of the `Trio` consists of the names of the sandwich, salad, and drink, in that order, each separated by `" / "` and followed by a space and then `"Trio"`. The price of the `Trio` is the sum of the two highest-priced items in the `Trio`; one item with the lowest price is free.

A trio consisting of a cheeseburger, spinach salad, and an orange soda would have the name `"Cheeseburger/Spinach Salad/Orange Soda Trio"` and a price of \$4.00 (the two highest prices are \$2.75 and \$1.25). Similarly, a trio consisting of a club sandwich, coleslaw, and a cappuccino would have the name `"Club Sandwich/Coleslaw/Cappuccino Trio"` and a price of \$6.25 (the two highest prices are \$2.75 and \$3.50).

Write the `Trio` class that implements the `MenuItem` interface. Your implementation must include a constructor that takes three parameters representing a sandwich, salad, and drink. The following code segment should have the indicated behavior.

```
Sandwich sandwich;  
Salad salad;  
Drink drink;  
/* Code that initializes sandwich, salad, and drink */  
  
Trio trio = new Trio(sandwich, salad, drink); // Compiles without error  
  
Trio trio1 = new Trio(salad, sandwich, drink); // Compile-time error  
Trio trio2 = new Trio(sandwich, salad, salad); // Compile-time error
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2014 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Write the complete `Trio` class below.



STOP

END OF EXAM



AP[®] Computer Science A 2013 Free-Response Questions

About the College Board

The College Board is a mission-driven not-for-profit organization that connects students to college success and opportunity. Founded in 1900, the College Board was created to expand access to higher education. Today, the membership association is made up of more than 6,000 of the world's leading educational institutions and is dedicated to promoting excellence and equity in education. Each year, the College Board helps more than seven million students prepare for a successful transition to college through programs and services in college readiness and college success — including the SAT[®] and the Advanced Placement Program[®]. The organization also serves the education community through research and advocacy on behalf of students, educators, and schools.

© 2013 The College Board. College Board, Advanced Placement Program, AP, AP Central, SAT, and the acorn logo are registered trademarks of the College Board. Admitted Class Evaluation Service and inspiring minds are trademarks owned by the College Board. All other products and services may be trademarks of their respective owners. Visit the College Board on the Web: www.collegeboard.org. Permission to use copyrighted College Board materials may be requested online at: www.collegeboard.org/inquiry/cbpermit.html.

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

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



2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

COMPUTER SCIENCE A SECTION II

Time—1 hour and 45 minutes

Number of questions—4

Percent of total score—50

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the appendices have been imported where appropriate.
 - Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
 - In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.
1. A music Web site keeps track of downloaded music. For each download, the site uses a `DownloadInfo` object to store a song's title and the number of times it has been downloaded. A partial declaration for the `DownloadInfo` class is shown below.

```
public class DownloadInfo
{
    /** Creates a new instance with the given unique title and sets the
     *   number of times downloaded to 1.
     *   @param title the unique title of the downloaded song
     */
    public DownloadInfo(String title)
    { /* implementation not shown */ }

    /** @return the title */
    public String getTitle()
    { /* implementation not shown */ }

    /** Increment the number times downloaded by 1 */
    public void incrementTimesDownloaded()
    { /* implementation not shown */ }

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

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

The list of downloaded information is stored in a `MusicDownloads` object. A partial declaration for the `MusicDownloads` class is shown below.

```
public class MusicDownloads
{
    /** The list of downloaded information.
     *  Guaranteed not to be null and not to contain duplicate titles.
     */
    private List<DownloadInfo> downloadList;

    /** Creates the list of downloaded information. */
    public MusicDownloads()
    { downloadList = new ArrayList<DownloadInfo>(); }

    /** Returns a reference to the DownloadInfo object with the requested title if it exists.
     *  @param title the requested title
     *  @return a reference to the DownloadInfo object with the
     *          title that matches the parameter title if it exists in the list;
     *          null otherwise.
     *  Postcondition:
     *  - no changes were made to downloadList.
     */
    public DownloadInfo getDownloadInfo(String title)
    { /* to be implemented in part (a) */ }

    /** Updates downloadList with information from titles.
     *  @param titles a list of song titles
     *  Postcondition:
     *  - there are no duplicate titles in downloadList.
     *  - no entries were removed from downloadList.
     *  - all songs in titles are represented in downloadList.
     *  - for each existing entry in downloadList, the download count is increased by
     *    the number of times its title appeared in titles.
     *  - the order of the existing entries in downloadList is not changed.
     *  - the first time an object with a title from titles is added to downloadList, it
     *    is added to the end of the list.
     *  - new entries in downloadList appear in the same order
     *    in which they first appear in titles.
     *  - for each new entry in downloadList, the download count is equal to
     *    the number of times its title appeared in titles.
     */
    public void updateDownloads(List<String> titles)
    { /* to be implemented in part (b) */ }

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

Part (a) begins on page 4.

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the `MusicDownloads` method `getDownloadInfo`, which returns a reference to a `DownloadInfo` object if an object with a title that matches the parameter `title` exists in the `downloadList`. If no song in `downloadList` has a title that matches the parameter `title`, the method returns `null`.

For example, suppose variable `webMusicA` refers to an instance of `MusicDownloads` and that the table below represents the contents of `downloadList`. The list contains three `DownloadInfo` objects. The object at position 0 has a title of "Hey Jude" and a download count of 5. The object at position 1 has a title of "Soul Sister" and a download count of 3. The object at position 2 has a title of "Aqualung" and a download count of 10.

0	1	2
"Hey Jude" 5	"Soul Sister" 3	"Aqualung" 10

The call `webMusicA.getDownloadInfo("Aqualung")` returns a reference to the object in position 2 of the list.

The call `webMusicA.getDownloadInfo("Happy Birthday")` returns `null` because there are no `DownloadInfo` objects with that title in the list.

Class information repeated from the beginning of the question

```
public class DownloadInfo

public DownloadInfo(String title)
public String getTitle()
public void incrementTimesDownloaded()

public class MusicDownloads

private List<DownloadInfo> downloadList
public DownloadInfo getDownloadInfo(String title)
public void updateDownloads(List<String> titles)
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `getDownloadInfo` below.

```
/** Returns a reference to the DownloadInfo object with the requested title if it exists.
 * @param title the requested title
 * @return a reference to the DownloadInfo object with the
 *         title that matches the parameter title if it exists in the list;
 *         null otherwise.
 * Postcondition:
 * - no changes were made to downloadList.
 */
public DownloadInfo getDownloadInfo(String title)
```



Part (b) begins on page 6.

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `MusicDownloads` method `updateDownloads`, which takes a list of song titles as a parameter. For each title in the list, the method updates `downloadList`, either by incrementing the download count if a `DownloadInfo` object with the same title exists, or by adding a new `DownloadInfo` object with that title and a download count of 1 to the end of the list. When a new `DownloadInfo` object is added to the end of the list, the order of the already existing entries in `downloadList` remains unchanged.

For example, suppose variable `webMusicB` refers to an instance of `MusicDownloads` and that the table below represents the contents of the instance variable `downloadList`.

0	1	2
"Hey Jude" 5	"Soul Sister" 3	"Aqualung" 10

Assume that the variable `List<String> songTitles` has been defined and contains the following entries.

```
{ "Lights", "Aqualung", "Soul Sister", "Go Now", "Lights", "Soul Sister" }
```

The call `webMusicB.updateDownloads(songTitles)` results in the following `downloadList` with incremented download counts for the objects with titles of "Soul Sister" and "Aqualung". It also has a new `DownloadInfo` object with a title of "Lights" and a download count of 2, and another `DownloadInfo` object with a title of "Go Now" and a download count of 1. The order of the already existing entries remains unchanged.

0	1	2	3	4
"Hey Jude" 5	"Soul Sister" 5	"Aqualung" 11	"Lights" 2	"Go Now" 1

Class information repeated from the beginning of the question

```
public class DownloadInfo  
  
public DownloadInfo(String title)  
public String getTitle()  
public void incrementTimesDownloaded()  
  
public class MusicDownloads  
  
private List<DownloadInfo> downloadList  
public DownloadInfo getDownloadInfo(String title)  
public void updateDownloads(List<String> titles)
```

In writing your solution, you must use the `getDownloadInfo` method. Assume that `getDownloadInfo` works as specified, regardless of what you wrote for part (a).

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `updateDownloads` below.

```
/** Updates downloadList with information from titles.
 * @param titles a list of song titles
 * Postcondition:
 *   - there are no duplicate titles in downloadList.
 *   - no entries were removed from downloadList.
 *   - all songs in titles are represented in downloadList.
 *   - for each existing entry in downloadList, the download count is increased by
 *     the number of times its title appeared in titles.
 *   - the order of the existing entries in downloadList is not changed.
 *   - the first time an object with a title from titles is added to downloadList, it
 *     is added to the end of the list.
 *   - new entries in downloadList appear in the same order
 *     in which they first appear in titles.
 *   - for each new entry in downloadList, the download count is equal to
 *     the number of times its title appeared in titles.
 */
public void updateDownloads(List<String> titles )
```

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

2. A multiplayer game called Token Pass has the following rules.

Each player begins with a random number of tokens (at least 1, but no more than 10) that are placed on a linear game board. There is one position on the game board for each player. After the game board has been filled, a player is randomly chosen to begin the game. Each position on the board is numbered, starting with 0.

The following rules apply for a player's turn.

- The tokens are collected and removed from the game board at that player's position.
- The collected tokens are distributed one at a time, to each player, beginning with the next player in order of increasing position.
- If there are still tokens to distribute after the player at the highest position gets a token, the next token will be distributed to the player at position 0.
- The distribution of tokens continues until there are no more tokens to distribute.

The Token Pass game board is represented by an array of integers. The indexes of the array represent the player positions on the game board, and the corresponding values in the array represent the number of tokens that each player has. The following example illustrates one player's turn.

Example

The following represents a game with 4 players. The player at position 2 was chosen to go first.

	0	1	2	3
Player Tokens	3	2	6	10

The tokens at position 2 are collected and distributed as follows.

- 1st token - to position 3 (The highest position is reached, so the next token goes to position 0.)
- 2nd token - to position 0
- 3rd token - to position 1
- 4th token - to position 2
- 5th token - to position 3 (The highest position is reached, so the next token goes to position 0.)
- 6th token - to position 0

After player 2's turn, the values in the array will be as follows.

	0	1	2	3
Player Tokens	5	3	1	12

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

The Token Pass game is represented by the `TokenPass` class.

```
public class TokenPass
{
    private int[] board;
    private int currentPlayer;

    /** Creates the board array to be of size playerCount and fills it with
     * random integer values from 1 to 10, inclusive. Initializes currentPlayer to a
     * random integer value in the range between 0 and playerCount-1, inclusive.
     * @param playerCount the number of players
     */
    public TokenPass(int playerCount)
    { /* to be implemented in part (a) */ }

    /** Distributes the tokens from the current player's position one at a time to each player in
     * the game. Distribution begins with the next position and continues until all the tokens
     * have been distributed. If there are still tokens to distribute when the player at the
     * highest position is reached, the next token will be distributed to the player at position 0.
     * Precondition: the current player has at least one token.
     * Postcondition: the current player has not changed.
     */
    public void distributeCurrentPlayerTokens()
    { /* to be implemented in part (b) */ }

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


2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the constructor for the `TokenPass` class. The parameter `playerCount` represents the number of players in the game. The constructor should create the `board` array to contain `playerCount` elements and fill the array with random numbers between 1 and 10, inclusive. The constructor should also initialize the instance variable `currentPlayer` to a random number between 0 and `playerCount-1`, inclusive.

Complete the `TokenPass` constructor below.

```
/** Creates the board array to be of size playerCount and fills it with
 * random integer values from 1 to 10, inclusive. Initializes currentPlayer to a
 * random integer value in the range between 0 and playerCount-1, inclusive.
 * @param playerCount the number of players
 */
public TokenPass(int playerCount)
```



Part (b) begins on page 11.

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `distributeCurrentPlayerTokens` method.

The tokens are collected and removed from the game board at the current player's position. These tokens are distributed, one at a time, to each player, beginning with the next higher position, until there are no more tokens to distribute.

Class information repeated from the beginning of the question

```
public class TokenPass

private int[] board
private int currentPlayer
public TokenPass(int playerCount)
public void distributeCurrentPlayerTokens()
```

Complete method `distributeCurrentPlayerTokens` below.

```
/** Distributes the tokens from the current player's position one at a time to each player in
 * the game. Distribution begins with the next position and continues until all the tokens
 * have been distributed. If there are still tokens to distribute when the player at the
 * highest position is reached, the next token will be distributed to the player at position 0.
 * Precondition: the current player has at least one token.
 * Postcondition: the current player has not changed.
 */
public void distributeCurrentPlayerTokens()
```

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

3. This question involves reasoning about the GridWorld case study. Reference materials are provided in the appendixes. In part (a) you will write a method to return an array list of all empty locations in a given grid. In part (b) you will write the class for a new type of `Critter`.

- (a) The `GridWorldUtilities` class contains static methods. A partial declaration of the `GridWorldUtilities` class is shown below.

```
public class GridWorldUtilities
{
    /** Gets all the locations in grid that do not contain objects.
     * @param grid a reference to a BoundedGrid object
     * @return an array list (possibly empty) of empty locations in grid.
     *         The size of the returned list is 0 if there are no empty locations in grid.
     *         Each empty location in grid should appear exactly once in the returned list.
     */
    public static ArrayList<Location> getEmptyLocations(Grid<Actor> grid)
    { /* to be implemented in part (a) */ }

    // There may be instance variables that are not shown.
}
```

Write the `GridWorldUtilities` method `getEmptyLocations`. If there are no empty locations in `grid`, the method returns an empty array list. Otherwise, it returns an array list of all empty locations in `grid`. Each empty location should appear exactly once in the array list.

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `getEmptyLocations` below.

```
/** Gets all the locations in grid that do not contain objects.
 * @param grid a reference to a BoundedGrid object
 * @return an array list (possibly empty) of empty locations in grid.
 *         The size of the returned list is 0 if there are no empty locations in grid.
 *         Each empty location in grid should appear exactly once in the returned list.
 */
public static ArrayList<Location> getEmptyLocations(Grid<Actor> grid)
```



Part (b) begins on page 14.

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) A `JumpingCritter` acts like a `Critter`, except that it moves by jumping to a randomly selected empty location in its grid. If there are no empty locations, the `JumpingCritter` removes itself from the grid.

The following diagram shows an example of a jumping critter that is able to move to an empty location. Example World #1 is shown below on the left. After the jumping critter at location (2, 0) acts, the world shown below on the right is one possible result.

EXAMPLE WORLD #1	POSSIBLE WORLD AFTER ACT
A jumping critter is in location (2, 0).	The jumping critter has eaten the bug that was in location (3, 1) and has moved to location (1, 3).

Example World #2 is shown below on the left. After the jumping critter at location (1, 2) acts, the world shown below on the right is the result.

EXAMPLE WORLD #2	WORLD AFTER ACT
A jumping critter is in location (1, 2).	The jumping critter removed itself from the grid because no empty locations were available.

Class information repeated from the beginning of the question

```
public class GridWorldUtilities
```

```
public static ArrayList<Location> getEmptyLocations(Grid<Actor> grid)
```

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

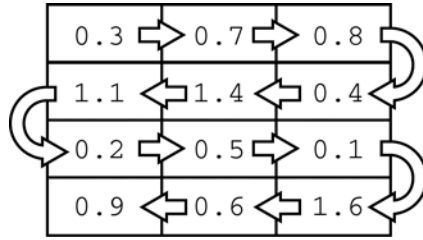
Assume that the `GridWorldUtilities` `getEmptyLocations` method works as specified, regardless of what you wrote in part (a). Solutions that reimplement the functionality of this method will not receive full credit.

Write the complete `JumpingCritter` class. Do NOT override the `act` method. Remember that your design must not violate the postconditions of the methods of the `Critter` class.



2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

4. A telescope scans a rectangular area of the night sky and collects the data into a 1-dimensional array. Each data value scanned is a number representing the amount of light detected by the telescope. The telescope scans back and forth across the sky (alternating between left to right and right to left) in the pattern indicated below by the arrows. The back-and-forth ordering of the values received from the scan is called *telescope order*.



The telescope records the data in telescope order into a 1-dimensional array of `double` values. This 1-dimensional array of information received from a single scan will be transferred into a 2-dimensional array, which reconstructs the original view of the rectangular area of the sky. This 2-dimensional array is part of the `SkyView` class, shown below. In this question you will write a constructor and a method for this class.

```
public class SkyView
{
    /** A rectangular array that holds the data representing a rectangular area of the sky. */
    private double[][] view;

    /** Constructs a SkyView object from a 1-dimensional array of scan data.
     * @param numRows the number of rows represented in the view
     * Precondition: numRows > 0
     * @param numCols the number of columns represented in the view
     * Precondition: numCols > 0
     * @param scanned the scan data received from the telescope, stored in telescope order
     * Precondition: scanned.length == numRows * numCols
     * Postcondition: view has been created as a rectangular 2-dimensional array
     *                    with numRows rows and numCols columns and the values in
     *                    scanned have been copied to view and are ordered as
     *                    in the original rectangular area of sky.
     */
    public SkyView(int numRows, int numCols, double[] scanned)
    { /* to be implemented in part (a) */ }

    /** Returns the average of the values in a rectangular section of view.
     * @param startRow the first row index of the section
     * @param endRow the last row index of the section
     * @param startCol the first column index of the section
     * @param endCol the last column index of the section
     * Precondition: 0 <= startRow <= endRow < view.length
     * Precondition: 0 <= startCol <= endCol < view[0].length
     * @return the average of the values in the specified section of view
     */
    public double getAverage(int startRow, int endRow,
                             int startCol, int endCol)
    { /* to be implemented in part (b) */ }

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

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the constructor for the `SkyView` class. The constructor initializes the `view` instance variable to a 2-dimensional array with `numRows` rows and `numCols` columns. The information from `scanned`, which is stored in the telescope order, is copied into `view` to reconstruct the sky view as originally seen by the telescope. The information in `scanned` must be rearranged as it is stored into `view` so that the sky view is oriented properly.

For example, suppose `scanned` contains values, as shown in the following array.

	0	1	2	3	4	5	6	7	8	9	10	11
scanned	0.3	0.7	0.8	0.4	1.4	1.1	0.2	0.5	0.1	1.6	0.6	0.9

Using the `scanned` array above, a `SkyView` object created with `new SkyView(4, 3, scanned)`, would have `view` initialized with the following values.

view	0	1	2
0	0.3	0.7	0.8
1	1.1	1.4	0.4
2	0.2	0.5	0.1
3	0.9	0.6	1.6

For another example, suppose `scanned` contains the following values.

	0	1	2	3	4	5
scanned	0.3	0.7	0.8	0.4	1.4	1.1

A `SkyView` object created with `new SkyView(3, 2, scanned)`, would have `view` initialized with the following values.

view	0	1
0	0.3	0.7
1	0.4	0.8
2	1.4	1.1

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete the `SkyView` constructor below.

```
/** Constructs a SkyView object from a 1-dimensional array of scan data.
 * @param numRows the number of rows represented in the view
 *      Precondition: numRows > 0
 * @param numCols the number of columns represented in the view
 *      Precondition: numCols > 0
 * @param scanned the scan data received from the telescope, stored in telescope order
 *      Precondition: scanned.length == numRows * numCols
 *      Postcondition: view has been created as a rectangular 2-dimensional array
 *                        with numRows rows and numCols columns and the values in
 *                        scanned have been copied to view and are ordered as
 *                        in the original rectangular area of sky.
 */
public SkyView(int numRows, int numCols, double[] scanned)
```



Part (b) begins on page 19.

© 2013 The College Board.
Visit the College Board on the Web: www.collegeboard.org.

GO ON TO THE NEXT PAGE.

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `SkyView` method `getAverage`, which returns the average of the elements of the section of `view` with row indexes from `startRow` through `endRow`, inclusive, and column indexes from `startCol` through `endCol`, inclusive.

For example, if `nightSky` is a `SkyView` object where `view` contains the values shown below, the call `nightSky.getAverage(1, 2, 0, 1)` should return `0.8`. (The average is $(1.1 + 1.4 + 0.2 + 0.5) / 4$, which equals `0.8`). The section being averaged is indicated by the dark outline in the table below.

view	0	1	2
0	0.3	0.7	0.8
1	1.1	1.4	0.4
2	0.2	0.5	0.1
3	0.9	0.6	1.6

Class information repeated from the beginning of the question

```
public class SkyView  
  
private double[][] view  
public SkyView(int numRows, int numCols, double[] scanned)  
public double getAverage(int startRow, int endRow,  
                        int startCol, int endCol)
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2013 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `getAverage` below.

```
/** Returns the average of the values in a rectangular section of view.
 * @param startRow the first row index of the section
 * @param endRow the last row index of the section
 * @param startCol the first column index of the section
 * @param endCol the last column index of the section
 * Precondition: 0 <= startRow <= endRow < view.length
 * Precondition: 0 <= startCol <= endCol < view[0].length
 * @return the average of the values in the specified section of view
 */
public double getAverage(int startRow, int endRow,
                        int startCol, int endCol)
```



STOP

END OF EXAM



AP[®] Computer Science A 2012 Free-Response Questions

About the College Board

The College Board is a mission-driven not-for-profit organization that connects students to college success and opportunity. Founded in 1900, the College Board was created to expand access to higher education. Today, the membership association is made up of more than 5,900 of the world's leading educational institutions and is dedicated to promoting excellence and equity in education. Each year, the College Board helps more than seven million students prepare for a successful transition to college through programs and services in college readiness and college success — including the SAT[®] and the Advanced Placement Program[®]. The organization also serves the education community through research and advocacy on behalf of students, educators, and schools.

© 2012 The College Board. College Board, Advanced Placement Program, AP, AP Central, SAT, and the acorn logo are registered trademarks of the College Board. Admitted Class Evaluation Service and inspiring minds are trademarks owned by the College Board. All other products and services may be trademarks of their respective owners. Visit the College Board on the Web: www.collegeboard.org. Permission to use copyrighted College Board materials may be requested online at: www.collegeboard.org/inquiry/cbpermit.html.

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

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



2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

COMPUTER SCIENCE A SECTION II

Time—1 hour and 45 minutes

Number of questions—4

Percent of total score—50

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the appendices have been imported where appropriate.
 - Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
 - In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.
1. A mountain climbing club maintains a record of the climbs that its members have made. Information about a climb includes the name of the mountain peak and the amount of time it took to reach the top. The information is contained in the `ClimbInfo` class as declared below.

```
public class ClimbInfo
{
    /** Creates a ClimbInfo object with name peakName and time climbTime.
     *  @param peakName the name of the mountain peak
     *  @param climbTime the number of minutes taken to complete the climb
     */
    public ClimbInfo(String peakName, int climbTime)
    { /* implementation not shown */ }

    /** @return the name of the mountain peak
     */
    public String getName()
    { /* implementation not shown */ }

    /** @return the number of minutes taken to complete the climb
     */
    public int getTime()
    { /* implementation not shown */ }

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

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

The `ClimbingClub` class maintains a list of the climbs made by members of the club. The declaration of the `ClimbingClub` class is shown below. You will write two different implementations of the `addClimb` method. You will also answer two questions about an implementation of the `distinctPeakNames` method.

```
public class ClimbingClub
{
    /** The list of climbs completed by members of the club.
     *   Guaranteed not to be null. Contains only non-null references.
     */
    private List<ClimbInfo> climbList;

    /** Creates a new ClimbingClub object. */
    public ClimbingClub()
    { climbList = new ArrayList<ClimbInfo>(); }

    /** Adds a new climb with name peakName and time climbTime to the list of climbs.
     *   @param peakName the name of the mountain peak climbed
     *   @param climbTime the number of minutes taken to complete the climb
     */
    public void addClimb(String peakName, int climbTime)
    { /* to be implemented in part (a) with ClimbInfo objects in the order they were added */
      /* to be implemented in part (b) with ClimbInfo objects in alphabetical order by name */
    }

    /** @return the number of distinct names in the list of climbs */
    public int distinctPeakNames()
    { /* implementation shown in part (c) */ }

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

Part (a) begins on page 4.

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write an implementation of the `ClimbingClub` method `addClimb` that stores the `ClimbInfo` objects in the order they were added. This implementation of `addClimb` should create a new `ClimbInfo` object with the given name and time. It appends a reference to that object to the end of `climbList`. For example, consider the following code segment.

```
ClimbingClub hikerClub = new ClimbingClub();
hikerClub.addClimb("Monadnock", 274);
hikerClub.addClimb("Whiteface", 301);
hikerClub.addClimb("Algonquin", 225);
hikerClub.addClimb("Monadnock", 344);
```

When the code segment has completed executing, the instance variable `climbList` would contain the following entries.

Peak Name	"Monadnock"	"Whiteface"	"Algonquin"	"Monadnock"
Climb Time	274	301	225	344

Information repeated from the beginning of the question

```
public class ClimbInfo
```

```
public ClimbInfo(String peakName, int climbTime)
public String getName()
public int getTime()
```

```
public class ClimbingClub
```

```
private List<ClimbInfo> climbList
public void addClimb(String peakName, int climbTime)
public int distinctPeakNames()
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `addClimb` below.

```
/** Adds a new climb with name peakName and time climbTime to the list of climbs.
 * @param peakName the name of the mountain peak climbed
 * @param climbTime the number of minutes taken to complete the climb
 * Postcondition: The new entry is at the end of climbList;
 *                 The order of the remaining entries is unchanged.
 */
public void addClimb(String peakName, int climbTime)
```



Part (b) begins on page 6.

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write an implementation of the `ClimbingClub` method `addClimb` that stores the elements of `climbList` in alphabetical order by name (as determined by the `compareTo` method of the `String` class). This implementation of `addClimb` should create a new `ClimbInfo` object with the given name and time and then insert the object into the appropriate position in `climbList`. Entries that have the same name will be grouped together and can appear in any order within the group. For example, consider the following code segment.

```
ClimbingClub hikerClub = new ClimbingClub();
hikerClub.addClimb("Monadnock", 274);
hikerClub.addClimb("Whiteface", 301);
hikerClub.addClimb("Algonquin", 225);
hikerClub.addClimb("Monadnock", 344);
```

When the code segment has completed execution, the instance variable `climbList` would contain the following entries in either of the orders shown below.

Peak Name	"Algonquin"	"Monadnock"	"Monadnock"	"Whiteface"
Climb Time	225	344	274	301

OR

Peak Name	"Algonquin"	"Monadnock"	"Monadnock"	"Whiteface"
Climb Time	225	274	344	301

You may assume that `climbList` is in alphabetical order by name when the method is called. When the method has completed execution, `climbList` should still be in alphabetical order by name.

Information repeated from the beginning of the question

```
public class ClimbInfo
```

```
public ClimbInfo(String peakName, int climbTime)
public String getName()
public int getTime()
```

```
public class ClimbingClub
```

```
private List<ClimbInfo> climbList
public void addClimb(String peakName, int climbTime)
public int distinctPeakNames()
```

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `addClimb` below.

```
/** Adds a new climb with name peakName and time climbTime to the list of climbs.  
 * Alphabetical order is determined by the compareTo method of the String class.  
 * @param peakName the name of the mountain peak climbed  
 * @param climbTime the number of minutes taken to complete the climb  
 * Precondition: entries in climbList are in alphabetical order by name.  
 * Postcondition: entries in climbList are in alphabetical order by name.  
 */  
public void addClimb(String peakName, int climbTime)
```



Part (c) begins on page 8.

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (c) The `ClimbingClub` method `distinctPeakNames` is intended to return the number of different names in `climbList`. For example, after the following code segment has completed execution, the value of the variable `numNames` would be 3.

```
ClimbingClub hikerClub = new ClimbingClub();
hikerClub.addClimb("Monadnock", 274);
hikerClub.addClimb("Whiteface", 301);
hikerClub.addClimb("Algonquin", 225);
hikerClub.addClimb("Monadnock", 344);
int numNames = hikerClub.distinctPeakNames();
```

Consider the following implementation of method `distinctPeakNames`.

```
/** @return the number of distinct names in the list of climbs */
public int distinctPeakNames()
{
    if (climbList.size() == 0)
    {
        return 0;
    }

    ClimbInfo currInfo = climbList.get(0);
    String prevName = currInfo.getName();
    String currName = null;
    int numNames = 1;

    for (int k = 1; k < climbList.size(); k++)
    {
        currInfo = climbList.get(k);
        currName = currInfo.getName();
        if (prevName.compareTo(currName) != 0)
        {
            numNames++;
            prevName = currName;
        }
    }
    return numNames;
}
```

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Assume that `addClimb` works as specified, regardless of what you wrote in parts (a) and (b).

- (i) Does this implementation of the `distinctPeakNames` method work as intended when the `addClimb` method stores the `ClimbInfo` objects in the order they were added as described in part (a)?

Circle one of the answers below.

YES

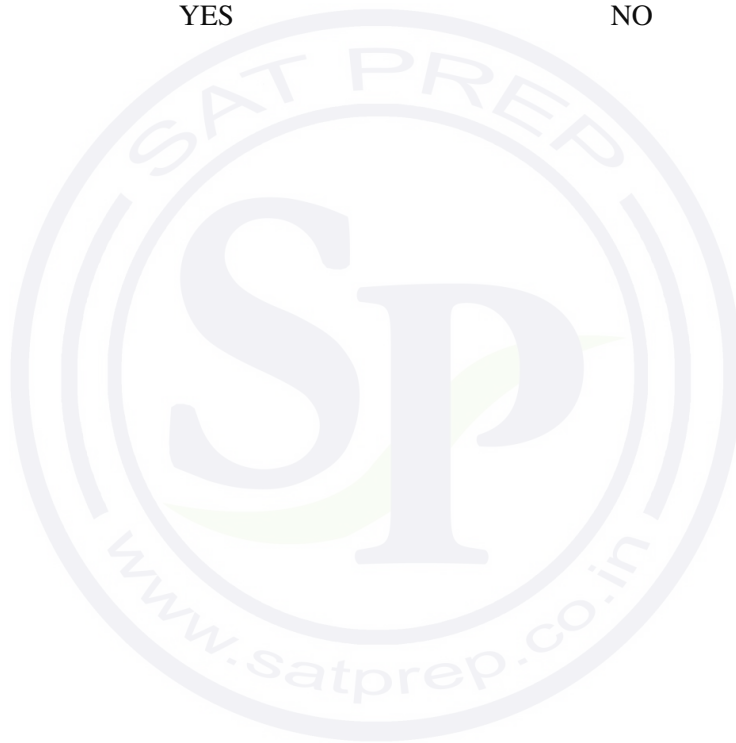
NO

- (ii) Does this implementation of the `distinctPeakNames` method work as intended when the `addClimb` method stores the `ClimbInfo` objects in alphabetical order by name as described in part (b)?

Circle one of the answers below.

YES

NO



2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

2. This question involves reasoning about the GridWorld case study. Reference materials are provided in the appendices.

A retro bug behaves like a regular bug. It also has the ability to revert to its previous location and direction. When a retro bug acts, it maintains information about its location and direction at the beginning of the act. The retro bug has a `restore` method that restores it to the location (if possible) and direction it faced at the beginning of its previous act. A retro bug only maintains information about its most recent act; therefore, multiple calls to `restore` that occur before its next act will use the same information. The `restore` method has no effect if it is called before a retro bug's first act.

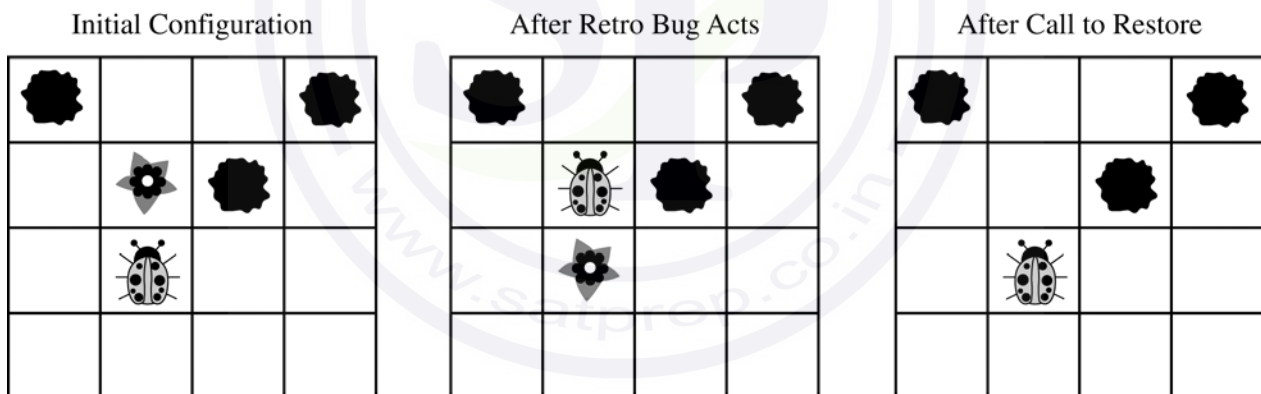
The `restore` method takes no parameters and does not return a value. The `restore` method has the following functionality.

- If the previous location of the retro bug is either unoccupied or contains a flower, the `restore` method places the retro bug in that previous location. The presence of any other type of actor in that location will prevent the retro bug from being placed in that location.
- The `restore` method always ends with the retro bug facing in the same direction that it had been facing at the beginning of its most recent act.

The following examples illustrate the behavior of the `restore` method.

Example 1

The retro bug acts once and later calls `restore`. Note that the flower that was originally in front of the retro bug is not replaced as a result of the call to `restore`. The retro bug is returned to its previous direction, which, in this case, is the same as the current direction.

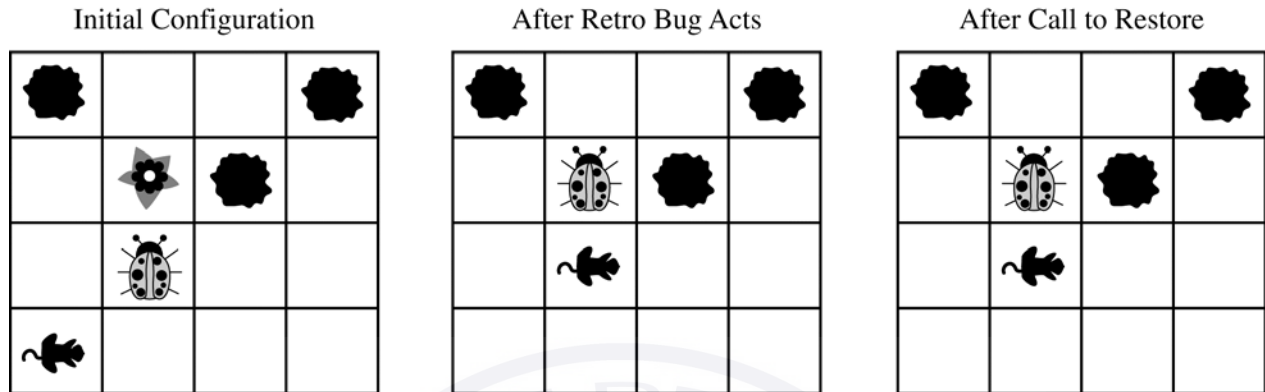


Question 2 continues on the next page.

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

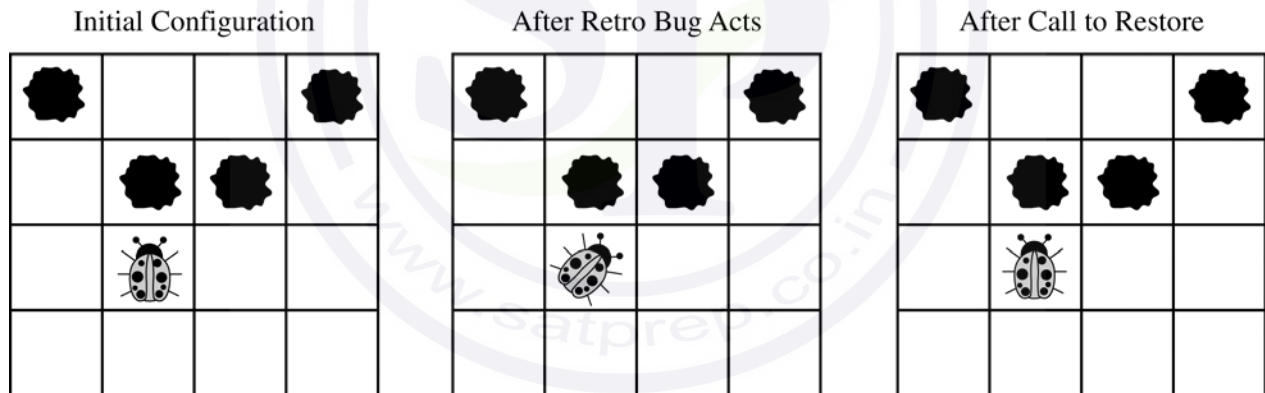
Example 2

The retro bug acts once and then some other actor moves into the location that the retro bug originally held. The call to `restore` results in the retro bug staying in its current location. The retro bug is returned to its previous direction (in this case it is the same as the current direction).



Example 3

The retro bug acts once and later calls `restore`. Because the retro bug is blocked from moving forward, it turns as its first act. The `restore` method results in the retro bug staying in its current location (the same as its previous location) and returning to its previous direction.



WRITE YOUR SOLUTION ON THE NEXT PAGE.

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Write the entire `RetroBug` class, including all necessary instance variables and methods.



2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

3. Consider a software system that models a horse barn. Classes that represent horses implement the following interface.

```
public interface Horse
{
    /** @return the horse's name */
    String getName();

    /** @return the horse's weight */
    int getWeight();

    // There may be methods that are not shown.
}
```

A horse barn consists of N numbered spaces. Each space can hold at most one horse. The spaces are indexed starting from 0; the index of the last space is $N - 1$. No two horses in the barn have the same name.

The declaration of the `HorseBarn` class is shown below. You will write two unrelated methods of the `HorseBarn` class.

```
public class HorseBarn
{
    /** The spaces in the barn. Each array element holds a reference to the horse
     *   that is currently occupying the space. A null value indicates an empty space.
     */
    private Horse[] spaces;

    /** Returns the index of the space that contains the horse with the specified name.
     *   Precondition: No two horses in the barn have the same name.
     *   @param name the name of the horse to find
     *   @return the index of the space containing the horse with the specified name;
     *           -1 if no horse with the specified name is in the barn.
     */
    public int findHorseSpace(String name)
    { /* to be implemented in part (a) */ }

    /** Consolidates the barn by moving horses so that the horses are in adjacent spaces,
     *   starting at index 0, with no empty space between any two horses.
     *   Postcondition: The order of the horses is the same as before the consolidation.
     */
    public void consolidate()
    { /* to be implemented in part (b) */ }

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

Part (a) begins on page 14.

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the `HorseBarn` method `findHorseSpace`. This method returns the index of the space in which the horse with the specified name is located. If there is no horse with the specified name in the barn, the method returns -1.

For example, assume a `HorseBarn` object called `sweetHome` has horses in the following spaces.

0	1	2	3	4	5	6
"Trigger" 1340	null	"Silver" 1210	"Lady" 1575	null	"Patches" 1350	"Duke" 1410

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

Method Call	Value Returned	Reason
<code>sweetHome.findHorseSpace("Trigger")</code>	0	A horse named Trigger is in space 0.
<code>sweetHome.findHorseSpace("Silver")</code>	2	A horse named Silver is in space 2.
<code>sweetHome.findHorseSpace("Coco")</code>	-1	A horse named Coco is not in the barn.

Information repeated from the beginning of the question

```
public interface Horse
```

```
String getName()  
int getWeight()
```

```
public class HorseBarn
```

```
private Horse[] spaces  
public int findHorseSpace(String name)  
public void consolidate()
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `findHorseSpace` below.

```
/** Returns the index of the space that contains the horse with the specified name.
 * Precondition: No two horses in the barn have the same name.
 * @param name the name of the horse to find
 * @return the index of the space containing the horse with the specified name;
 *         -1 if no horse with the specified name is in the barn.
 */
public int findHorseSpace(String name)
```



Part (b) begins on page 16.

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `HorseBarn` method `consolidate`. This method consolidates the barn by moving horses so that the horses are in adjacent spaces, starting at index 0, with no empty spaces between any two horses. After the barn is consolidated, the horses are in the same order as they were before the consolidation.

For example, assume a barn has horses in the following spaces.

0	1	2	3	4	5	6
"Trigger" 1340	null	"Silver" 1210	null	null	"Patches" 1350	"Duke" 1410

The following table shows the arrangement of the horses after `consolidate` is called.

0	1	2	3	4	5	6
"Trigger" 1340	"Silver" 1210	"Patches" 1350	"Duke" 1410	null	null	null

Information repeated from the beginning of the question

```
public interface Horse
```

```
String getName()  
int getWeight()
```

```
public class HorseBarn
```

```
private Horse[] spaces  
public int findHorseSpace(String name)  
public void consolidate()
```

WRITE YOUR SOLUTION ON THE NEXT PAGE.

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `consolidate` below.

```
/** Consolidates the barn by moving horses so that the horses are in adjacent spaces,  
 * starting at index 0, with no empty space between any two horses.  
 * Postcondition: The order of the horses is the same as before the consolidation.  
 */  
public void consolidate()
```



2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

4. A grayscale image is represented by a 2-dimensional rectangular array of pixels (picture elements). A pixel is an integer value that represents a shade of gray. In this question, pixel values can be in the range from 0 through 255, inclusive. A black pixel is represented by 0, and a white pixel is represented by 255.

The declaration of the `GrayImage` class is shown below. You will write two unrelated methods of the `GrayImage` class.

```
public class GrayImage
{
    public static final int BLACK = 0;
    public static final int WHITE = 255;

    /** The 2-dimensional representation of this image. Guaranteed not to be null.
     * All values in the array are within the range [BLACK, WHITE], inclusive.
     */
    private int[][] pixelValues;

    /** @return the total number of white pixels in this image.
     * Postcondition: this image has not been changed.
     */
    public int countWhitePixels()
    { /* to be implemented in part (a) */ }

    /** Processes this image in row-major order and decreases the value of each pixel at
     * position (row, col) by the value of the pixel at position (row + 2, col + 2) if it exists.
     * Resulting values that would be less than BLACK are replaced by BLACK.
     * Pixels for which there is no pixel at position (row + 2, col + 2) are unchanged.
     */
    public void processImage()
    { /* to be implemented in part (b) */ }
}
```

- (a) Write the method `countWhitePixels` that returns the number of pixels in the image that contain the value `WHITE`. For example, assume that `pixelValues` contains the following image.

	0	1	2	3	4
0	255	184	178	84	129
1	84	255	255	130	84
2	78	255	0	0	78
3	84	130	255	130	84

A call to `countWhitePixels` method would return 5 because there are 5 entries (shown in boldface) that have the value `WHITE`.

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `countWhitePixels` below.

```
/** @return the total number of white pixels in this image.  
 *   Postcondition: this image has not been changed.  
 */  
public int countWhitePixels()
```



Part (b) begins on page 20.

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the method `processImage` that modifies the image by changing the values in the instance variable `pixelValues` according to the following description. The pixels in the image are processed one at a time in row-major order. Row-major order processes the first row in the array from left to right and then processes the second row from left to right, continuing until all rows are processed from left to right. The first index of `pixelValues` represents the row number, and the second index represents the column number.

The pixel value at position (row, col) is decreased by the value at position (row + 2, col + 2) if such a position exists. If the result of the subtraction is less than the value `BLACK`, the pixel is assigned the value of `BLACK`. The values of the pixels for which there is no pixel at position (row + 2, col + 2) remain unchanged. You may assume that all the original values in the array are within the range [`BLACK`, `WHITE`], inclusive.

The following diagram shows the contents of the instance variable `pixelValues` before and after a call to `processImage`. The values shown in boldface represent the pixels that could be modified in a grayscale image with 4 rows and 5 columns.

Before Call to <code>processImage</code>						After Call to <code>processImage</code>					

2012 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

Complete method `processImage` below.

```
/** Processes this image in row-major order and decreases the value of each pixel at
 * position (row, col) by the value of the pixel at position (row + 2, col + 2) if it exists.
 * Resulting values that would be less than BLACK are replaced by BLACK.
 * Pixels for which there is no pixel at position (row + 2, col + 2) are unchanged.
 */
public void processImage()
```



STOP

END OF EXAM



AP[®] Computer Science A 2011 Free-Response Questions

About the College Board

The College Board is a mission-driven not-for-profit organization that connects students to college success and opportunity. Founded in 1900, the College Board was created to expand access to higher education. Today, the membership association is made up of more than 5,900 of the world's leading educational institutions and is dedicated to promoting excellence and equity in education. Each year, the College Board helps more than seven million students prepare for a successful transition to college through programs and services in college readiness and college success — including the SAT[®] and the Advanced Placement Program[®]. The organization also serves the education community through research and advocacy on behalf of students, educators and schools.

© 2011 The College Board. College Board, Advanced Placement Program, AP, AP Central, SAT and the acorn logo are registered trademarks of the College Board. Admitted Class Evaluation Service and inspiring minds are trademarks owned by the College Board. All other products and services may be trademarks of their respective owners. Visit the College Board on the Web: www.collegeboard.org. Permission to use copyrighted College Board materials may be requested online at: www.collegeboard.org/inquiry/cbpermit.html.

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

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

2011 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

COMPUTER SCIENCE A SECTION II

Time—1 hour and 45 minutes

Number of questions—4

Percent of total score—50

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Quick Reference found in the Appendix have been imported where appropriate.
 - Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
 - In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.
1. Digital sounds can be represented as an array of integer values. For this question, you will write two unrelated methods of the `Sound` class.

A partial declaration of the `Sound` class is shown below.

```
public class Sound
{
    /** the array of values in this sound; guaranteed not to be null */
    private int[] samples;

    /** Changes those values in this sound that have an amplitude greater than limit.
     * Values greater than limit are changed to limit.
     * Values less than -limit are changed to -limit.
     * @param limit the amplitude limit
     * Precondition: limit ≥ 0
     * @return the number of values in this sound that this method changed
     */
    public int limitAmplitude(int limit)
    { /* to be implemented in part (a) */ }

    /** Removes all silence from the beginning of this sound.
     * Silence is represented by a value of 0.
     * Precondition: samples contains at least one nonzero value
     * Postcondition: the length of samples reflects the removal of starting silence
     */
    public void trimSilenceFromBeginning()
    { /* to be implemented in part (b) */ }

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

2011 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) The volume of a sound depends on the amplitude of each value in the sound. The amplitude of a value is its absolute value. For example, the amplitude of -2300 is 2300 and the amplitude of 4000 is 4000.

Write the method `limitAmplitude` that will change any value that has an amplitude greater than the given `limit`. Values that are greater than `limit` are replaced with `limit`, and values that are less than `-limit` are replaced with `-limit`. The method returns the total number of values that were changed in the array. For example, assume that the array `samples` has been initialized with the following values.

40	2532	17	-2300	-17	-4000	2000	1048	-420	33	15	-32	2030	3223
----	------	----	-------	-----	-------	------	------	------	----	----	-----	------	------

When the statement

```
int numChanges = limitAmplitude(2000);
```

is executed, the value of `numChanges` will be 5, and the array `samples` will contain the following values.

40	2000	17	-2000	-17	-2000	2000	1048	-420	33	15	-32	2000	2000
----	------	----	-------	-----	-------	------	------	------	----	----	-----	------	------

Complete method `limitAmplitude` below.

```
/** Changes those values in this sound that have an amplitude greater than limit.
 * Values greater than limit are changed to limit.
 * Values less than -limit are changed to -limit.
 * @param limit the amplitude limit
 * Precondition: limit ≥ 0
 * @return the number of values in this sound that this method changed
 */
public int limitAmplitude(int limit)
```

2011 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Recorded sound often begins with silence. Silence in a sound is represented by a value of 0.

Write the method `trimSilenceFromBeginning` that removes the silence from the beginning of a sound. To remove starting silence, a new array of values is created that contains the same values as the original `samples` array in the same order but without the leading zeros. The instance variable `samples` is updated to refer to the new array. For example, suppose the instance variable `samples` refers to the following array.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value	0	0	0	0	-14	0	-35	-39	0	-7	16	32	37	29	0	0

After `trimSilenceFromBeginning` has been called, the instance variable `samples` will refer to the following array.

Index	0	1	2	3	4	5	6	7	8	9	10	11
Value	-14	0	-35	-39	0	-7	16	32	37	29	0	0

Complete method `trimSilenceFromBeginning` below.

```

/** Removes all silence from the beginning of this sound.
 * Silence is represented by a value of 0.
 * Precondition: samples contains at least one nonzero value
 * Postcondition: the length of samples reflects the removal of starting silence
 */
public void trimSilenceFromBeginning()

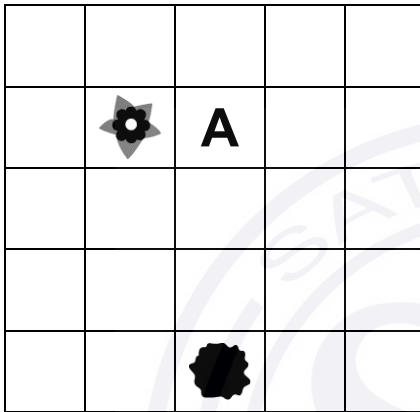
```

2011 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

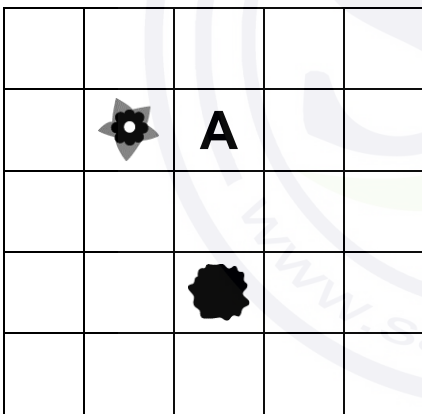
2. This question involves reasoning about the GridWorld case study. Reference materials are provided in the Appendix.

An attractive critter is a critter that processes other actors by attempting to relocate all of the other actors in the grid, including other attractive critters. The attractive critter attempts to move each other actor one grid cell closer to itself in the direction specified by `getDirectionToward`. An actor is relocated only if the location into which it would be relocated is empty. If an actor cannot be moved, it is left in its original position. After trying to move all other actors, the attractive critter moves like a critter.

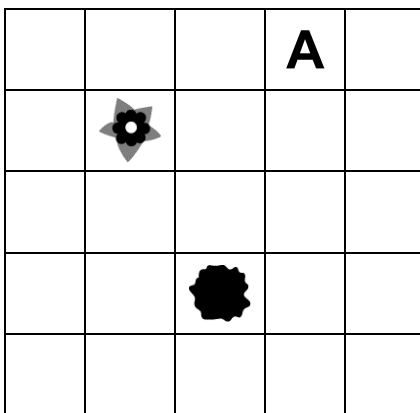
The following series of figures represents an example of how the attractive critter affects the other actors in the grid.



An attractive critter (indicated by the letter A) is in location (1, 2), a flower is in location (1, 1), and a rock is in location (4, 2).



When the attractive critter acts, the rock will be relocated to location (3, 2) because that location is one grid cell closer to the attractive critter and is empty. The flower will not be relocated because the grid cell that is one location closer to the attractive critter is already occupied.



After attempting to relocate all the other actors in the grid, the attractive critter then moves like a critter. In this example, the attractive critter moves to location (0, 3).

2011 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

The order in which the actors in the grid are processed is not specified, making it possible to get different results from the same grid of actors.

Write the complete `AttractiveCritic` class, including all instance variables and required methods. Do NOT override the `act` method. Remember that your design must not violate the postconditions of the methods of the `Critic` class and that updating an object's instance variable changes the state of that object.



2011 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

3. A fuel depot has a number of fuel tanks arranged in a line and a robot that moves a filling mechanism back and forth along the line so that the tanks can be filled. A fuel tank is specified by the `FuelTank` interface below.

```
public interface FuelTank
{
    /** @return an integer value that ranges from 0 (empty) to 100 (full) */
    int getFuelLevel();
}
```

A fuel depot keeps track of the fuel tanks and the robot. The following figure represents the tanks and the robot in a fuel depot. The robot, indicated by the arrow, is currently at index 2 and is facing to the right.

Tank index	0	1	2	3	4	5
Fuel level in tank	80	70	20	45	50	25
Robot	→					

The state of the robot includes the index of its location and the direction in which it is facing (to the right or to the left). This information is specified in the `FuelRobot` interface as shown in the following declaration.

```
public interface FuelRobot
{
    /** @return the index of the current location of the robot */
    int getCurrentIndex();

    /** Determine whether the robot is currently facing to the right
     *  @return true if the robot is facing to the right (toward tanks with larger indexes)
     *          false if the robot is facing to the left (toward tanks with smaller indexes)
     */
    boolean isFacingRight();

    /** Changes the current direction of the robot */
    void changeDirection();

    /** Moves the robot in its current direction by the number of locations specified.
     *  @param numLocs the number of locations to move. A value of 1 moves
     *               the robot to the next location in the current direction.
     *               Precondition: numLocs > 0
     */
    void moveForward(int numLocs);
}
```

2011 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

A fuel depot is represented by the `FuelDepot` class as shown in the following class declaration.

```
public class FuelDepot
{
    /** The robot used to move the filling mechanism */
    private FuelRobot filler;

    /** The list of fuel tanks */
    private List<FuelTank> tanks;

    /** Determines and returns the index of the next tank to be filled.
     *  @param threshold fuel tanks with a fuel level  $\leq$  threshold may be filled
     *  @return index of the location of the next tank to be filled
     *  Postcondition: the state of the robot has not changed
     */
    public int nextTankToFill(int threshold)
    { /* to be implemented in part (a) */ }

    /** Moves the robot to location locIndex.
     *  @param locIndex the index of the location of the tank to move to
     *  Precondition:  $0 \leq \text{locIndex} < \text{tanks.size}()$ 
     *  Postcondition: the current location of the robot is locIndex
     */
    public void moveToLocation(int locIndex)
    { /* to be implemented in part (b) */ }

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


2011 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the `FuelDepot` method `nextTankToFill` that returns the index of the next tank to be filled.

The index for the next tank to be filled is determined according to the following rules:

- Return the index of a tank with the lowest fuel level that is less than or equal to a given threshold.
If there is more than one fuel tank with the same lowest fuel level, any of their indexes can be returned.
- If there are no tanks with a fuel level less than or equal to the threshold, return the robot's current index.

For example, suppose the tanks contain the fuel levels shown in the following figure.

Tank index	0	1	2	3	4	5	6
Fuel level in tank	20	30	80	55	50	75	20
Robot	→						

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

threshold	Return Value	Rationale
50	0 or 6	20 is the lowest fuel level, so either 0 or 6 can be returned.
15	2	There are no tanks with a fuel level \leq threshold, so the robot's current index is returned.

Complete method `nextTankToFill` below.

```

/** Determines and returns the index of the next tank to be filled.
 * @param threshold fuel tanks with a fuel level  $\leq$  threshold may be filled
 * @return index of the location of the next tank to be filled
 * Postcondition: the state of the robot has not changed
 */
public int nextTankToFill(int threshold)

```

2011 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `FuelDepot` method `moveToLocation` that will move the robot to the given tank location. Because the robot can only move forward, it may be necessary to change the direction of the robot before having it move. Do not move the robot past the end of the line of fuel tanks.

Complete method `moveToLocation` below.

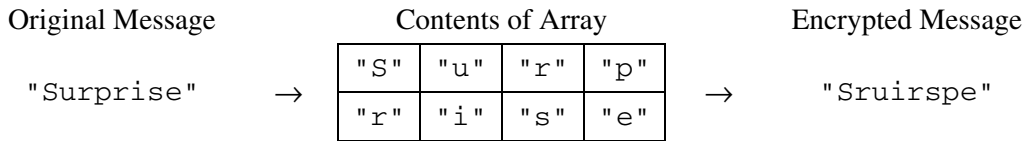
```
/** Moves the robot to location locIndex.
 * @param locIndex the index of the location of the tank to move to
 * Precondition:  $0 \leq \text{locIndex} < \text{tanks.size}()$ 
 * Postcondition: the current location of the robot is locIndex
 */
public void moveToLocation(int locIndex)
```



2011 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

4. In this question you will write two methods for a class `RouteCipher` that encrypts (puts into a coded form) a message by changing the order of the characters in the message. The route cipher fills a two-dimensional array with single-character substrings of the original message in row-major order, encrypting the message by retrieving the single-character substrings in column-major order.

For example, the word "Surprise" can be encrypted using a 2-row, 4-column array as follows.



An incomplete implementation of the `RouteCipher` class is shown below.

```
public class RouteCipher
{
    /** A two-dimensional array of single-character strings, instantiated in the constructor */
    private String[][] letterBlock;

    /** The number of rows of letterBlock, set by the constructor */
    private int numRows;

    /** The number of columns of letterBlock, set by the constructor */
    private int numCols;

    /** Places a string into letterBlock in row-major order.
     * @param str the string to be processed
     * Postcondition:
     *   if str.length() < numRows * numCols, "A" is placed in each unfilled cell
     *   if str.length() > numRows * numCols, trailing characters are ignored
     */
    private void fillBlock(String str)
    { /* to be implemented in part (a) */ }

    /** Extracts encrypted string from letterBlock in column-major order.
     * Precondition: letterBlock has been filled
     * @return the encrypted string from letterBlock
     */
    private String encryptBlock()
    { /* implementation not shown */ }

    /** Encrypts a message.
     * @param message the string to be encrypted
     * @return the encrypted message;
     *         if message is the empty string, returns the empty string
     */
    public String encryptMessage(String message)
    { /* to be implemented in part (b) */ }

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

2011 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the method `fillBlock` that fills the two-dimensional array `letterBlock` with one-character strings from the string passed as parameter `str`.

The array must be filled in row-major order—the first row is filled from left to right, then the second row is filled from left to right, and so on, until all rows are filled.

If the length of the parameter `str` is smaller than the number of elements of the array, the string "A" is placed in each of the unfilled cells. If the length of `str` is larger than the number of elements in the array, the trailing characters are ignored.

For example, if `letterBlock` has 3 rows and 5 columns and `str` is the string "Meet at noon", the resulting contents of `letterBlock` would be as shown in the following table.

"M"	"e"	"e"	"t"	" "
"a"	"t"	" "	"n"	"o"
"o"	"n"	"A"	"A"	"A"

If `letterBlock` has 3 rows and 5 columns and `str` is the string "Meet at midnight", the resulting contents of `letterBlock` would be as shown in the following table.

"M"	"e"	"e"	"t"	" "
"a"	"t"	" "	"m"	"i"
"d"	"n"	"i"	"g"	"h"

The following expression may be used to obtain a single-character string at position `k` of the string `str`.

```
str.substring(k, k + 1)
```

Complete method `fillBlock` below.

```
/** Places a string into letterBlock in row-major order.
 * @param str the string to be processed
 * Postcondition:
 *   if str.length() < numRows * numCols, "A" is placed in each unfilled cell
 *   if str.length() > numRows * numCols, trailing characters are ignored
 */
private void fillBlock(String str)
```

2011 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the method `encryptMessage` that encrypts its string parameter `message`. The method builds an encrypted version of `message` by repeatedly calling `fillBlock` with consecutive, nonoverlapping substrings of `message` and concatenating the results returned by a call to `encryptBlock` after each call to `fillBlock`. When all of `message` has been processed, the concatenated string is returned. Note that if `message` is the empty string, `encryptMessage` returns an empty string.

The following example shows the process carried out if `letterBlock` has 2 rows and 3 columns and `encryptMessage("Meet at midnight")` is executed.

Substring	letterBlock after Call to fillBlock	Value Returned by encryptBlock	Concatenated String						
"Meet a"	<table><tr><td>"M"</td><td>"e"</td><td>"e"</td></tr><tr><td>"t"</td><td>" "</td><td>"a"</td></tr></table>	"M"	"e"	"e"	"t"	" "	"a"	"Mte ea"	"Mte ea"
"M"	"e"	"e"							
"t"	" "	"a"							
"t midn"	<table><tr><td>"t"</td><td>" "</td><td>"m"</td></tr><tr><td>"i"</td><td>"d"</td><td>"n"</td></tr></table>	"t"	" "	"m"	"i"	"d"	"n"	"ti dmn"	"Mte eati dmn"
"t"	" "	"m"							
"i"	"d"	"n"							
"ight"	<table><tr><td>"i"</td><td>"g"</td><td>"h"</td></tr><tr><td>"t"</td><td>"A"</td><td>"A"</td></tr></table>	"i"	"g"	"h"	"t"	"A"	"A"	"itgAhA"	"Mte eati dmnitgAhA"
"i"	"g"	"h"							
"t"	"A"	"A"							

In this example, the method returns the string "Mte eati dmnitgAhA".

Assume that `fillBlock` and `encryptBlock` methods work as specified. Solutions that reimplement the functionality of one or both of these methods will not receive full credit.

Complete method `encryptMessage` below.

```
/** Encrypts a message.
 * @param message the string to be encrypted
 * @return the encrypted message;
 *         if message is the empty string, returns the empty string
 */
public String encryptMessage(String message)
```

STOP

END OF EXAM