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.

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.

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</pre>
     * /
   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;</pre>
     *
            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</pre>
     * /
   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</pre>
   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.

© 2023 College Board. Visit College Board on the web: collegeboard.org.

}

(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.

(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	

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.

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)
```

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	Everything on s ale, please com e in		
17	Everything on sal e, please come in		
40	Everything on sale, please come in		

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.

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
String str;	,	
int x;		
Sign sign1 = new		The message for sign1 contains 8
Sign("ABC222DE", 3);		characters, and the sign has lines of width 3.
		The sign needs three lines to display the
<pre>x = sign1.numberOfLines();</pre>	3	8-character message on a sign with lines of width 3.
<pre>str = sign1.getLines();</pre>	"ABC;222;DE"	Semicolons separate the text displayed on the first, second, and third lines of the sign.
<pre>str = sign1.getLines();</pre>	"ABC;222;DE"	Successive calls to getLines return the
		same value.
Sign sign2 = new		The message for sign2 contains 4
Sign("ABCD", 10);		characters, and the sign has lines of width 10.
		The sign needs one line to display the
x = sign2.numberOfLines();	1	4-character message on a sign with lines of
		width 10.
str = sign2.getLines();	"ABCD"	No semicolon appears, since the text to be
	71000	displayed fits on the first line of the sign.
Sign sign3 = new		The message for sign3 contains 6
Sign("ABCDEF", 6);		characters, and the sign has lines of width 6.
		The sign needs one line to display the
x = sign3.numberOfLines();	1	6-character message on a sign with lines of
		width 6.
<pre>str = sign3.getLines();</pre>	"ABCDEF"	No semicolon appears, since the text to be
	1100001	displayed fits on the first line of the sign.
<pre>Sign sign4 = new Sign("", 4);</pre>		The message for sign4 is an empty string.
x = sign4.numberOfLines();	0	There is no text to display.
<pre>str = sign4.getLines();</pre>	nul1	There is no text to display.
Sign sign5 = new		The message for sign5 contains 8
Sign("AB_CD_EF", 2);		characters, and the sign has lines of width 2.
		The sign needs four lines to display the
x = sign5.numberOfLines();	4	8-character message on a sign with lines of
		width 2.
str = sign5.getLines();	"AB;_C;D_;EF"	Semicolons separate the text displayed on the
SCI - Signo.gechines(//	11D, _C, D_, Hr	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.

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. 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.

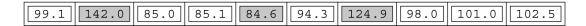
```
public class WeatherData
{
    /** Guaranteed not to be null and to contain only non-null entries */
   private ArrayList<Double> temperatures;
    /**
        Cleans the data by removing from temperatures all values that are less than
         lower and all values that are greater than upper, as 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
         Precondition: There is at least one heat wave in temperatures based on threshold.
     * /
   public int longestHeatWave(double threshold)
    \{ /* \text{ to be implemented in part (b) } */ \}
    // There may be instance variables, constructors, and methods that are not shown.
}
```

(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.

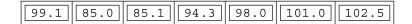
For example, consider a WeatherData object for which temperatures contains the following.



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



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).



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.

(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.

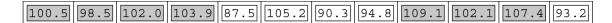
For example, consider the following contents of temperatures.



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.



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)
```

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)
    \{ /* \text{ 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)
    \{ /* \text{ to be implemented in part (b) } */ \}
    // There may be instance variables, constructors, and methods that are not shown.
```

- (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.

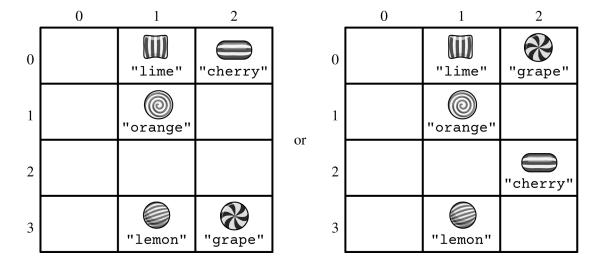
In the following example, the grid represents the contents of box. An empty square in the grid is null in box. A non-empty square in the grid represents a box element that contains a Candy object. The string in the square of the grid indicates the flavor of the piece of candy.

	0	1	2
0		W "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.

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.



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.

© 2023 College Board. Visit College Board on the web: collegeboard.org. (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	W "lime"	Wilime"		"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	Wilime"	"lime"		"lemon"	
1	orange"			"lime"	"lime"
2			"lemon"		orange"

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	Wlime"	Wlime"		"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)
```

AP® Computer Science A 2023 Free-Response Questions

STOP

END OF EXAM