CollegeBoard AP

INCLUDES
- Course framework
- Instructional section
- Sample exam questions

AP® Computer Science A

COURSE AND EXAM DESCRIPTION

Effective
Fall 2020

Sample Exam Questions

The sample exam questions that follow illustrate the relationship between the course framework and the AP Computer Science A Exam and serve as examples of the types of questions that appear on the exam. After the sample questions is a table that shows to which skill, learning objective(s), and unit each question relates. The table also provides the answers to the multiple-choice questions.

Section I: Multiple-Choice

1. Consider the following code segment.

```
int a = 5;
int b = 2;
double c = 3.0;
System.out.println(5 + a / b * c - 1);
```

What is printed when the code segment is executed?

(A) `0.666666666666667`

(B) `9.0`

(C) `10.0`

(D) `11.5`

(E) `14.0`

2. Consider the `processWords` method. Assume that each of its two parameters is a `String` of length two or more.

```
public void processWords(String word1, String word2)
{
    String str1 = word1.substring(0, 2);
    String str2 = word2.substring(word2.length() - 1);
    String result = str2 + str1;
    System.out.println(result.indexOf(str2));
}
```

Which of the following best describes the value printed when `processWords` is called?

(A) The value `0` is always printed.

(B) The value `1` is always printed.

(C) The value `result.length() - 1` is printed.

(D) A substring containing the last character of `word2` is printed.

(E) A substring containing the last two characters of `word2` is printed.

3. Which of the following statements assigns a random integer between 25 and 60, inclusive, to `rn`?

(A) `int rn = (int) (Math.random() * 25) + 36;`

(B) `int rn = (int) (Math.random() * 25) + 60;`

(C) `int rn = (int) (Math.random() * 26) + 60;`

(D) `int rn = (int) (Math.random() * 36) + 25;`

(E) `int rn = (int) (Math.random() * 60) + 25;`

4. Vehicles are classified based on their total interior volume. The `classify` method is intended to return a vehicle classification `String` value based on total interior volume, in cubic feet, as shown in the table below.

Vehicle size class	Total interior volume
Minicompact	Less than 85 cubic feet
Subcompact	85 to 99 cubic feet
Compact	100 to 109 cubic feet
Mid-Size	110 to 119 cubic feet
Large	120 cubic feet or more

The `classify` method, which does not work as intended, is shown below.

```
public static String classify(int volume)
{
   String carClass = "";
   if (volume >= 120)
   {
      carClass = "Large";
   }
   else if (volume < 120)
   {
      carClass = "Mid-Size";
   }
   else if (volume < 110)
   {
      carClass = "Compact";
   }
   else if (volume < 100)
   {
      carClass = "Subcompact";
   }
```

```
    else
    {
       carClass = "Minicompact";
    }
    return carClass;
}
```

The `classify` method works as intended for some but not all values of the parameter `volume`. For which of the following values of `volume` would the correct value be returned when the `classify` method is executed?

(A) `80`

(B) `90`

(C) `105`

(D) `109`

(E) `115`

5. Which of the following best describes the value of the Boolean expression shown below?

```
a && !(b || a)
```

(A) The value is always `true`.

(B) The value is always `false`.

(C) The value is `true` when `a` has the value `false`, and is `false` otherwise.

(D) The value is `true` when `b` has the value `false`, and is `false` otherwise.

(E) The value is `true` when either `a` or `b` has the value `true`, and is `false` otherwise.

6. Consider the following code segment.

```
int val = 48;
int div = 6;
while ((val % 2 == 0) && div > 0)
{
    if (val % div == 0)
    {
       System.out.print(val + " ");
    }
    val /= 2;
    div--;
}
```

What is printed when the code segment is executed?

(A) `48 12 6`

(B) `48 12 6 3`

(C) `48 12 6 3 1`

(D) `48 24 12 6`

(E) `48 24 12 6 3`

7. Consider the following class definition.

```
public class Example
{
   private int x;
   // Constructor not shown.
}
```

Which of the following is a correct header for a method of the `Example` class that would return the value of the `private` instance variable `x` so that it can be used in a class other than `Example`?

(A) `private int getX()`

(B) `private void getX()`

(C) `public int getX()`

(D) `public void getX()`

(E) `public void getX(int x)`

8. In the following code segment, assume that the string `str` has been properly declared and initialized. The code segment is intended to print the number of strings in the array `animals` that have `str` as a substring.

```
String[] animals = {"horse", "cow", "goat", "dog",
                    "cat", "mouse"};
int count = 0;
for (int i = 0; i <= animals.length; i++)
{
    if (animals[i].indexOf(str) >= 0)
    {
       count++;
    }
}
System.out.println(count);
```

Which of the following changes should be made so the code segment works as intended?

(A) The Boolean expression in the for loop header should be changed to `i < animals.length`.

(B) The Boolean expression in the for loop header should be changed to `i < animals.length -1`.

(C) The Boolean expression in the for loop header should be changed to `i < animals[i].length`.

(D) The condition in the `if` statement should be changed to `animals[i].equals(str)`.

(E) The condition in the `if` statement should be changed to `animals[i].substring(str)`.

9. Consider an integer array, `nums`, which has been declared and initialized with one or more integer values. Which of the following code segments updates `nums` so that each element contains the square of its original value?

I.

```
int k = 0;
while (k < nums.length)
{
   nums[k] = nums[k] * nums[k];
}
```

II.

```
for (int k = 0; k < nums.length; k++)
{
   nums[k] = nums[k] * nums[k];
}
```

III.

```
for (int n : nums)
{
   n = n * n;
}
```

(A) II only

(B) I and II only

(C) I and III only

(D) II and III only

(E) I, II, and III

10. Consider the following code segment.

```
ArrayList<String> numbers = new ArrayList<String>();
numbers.add("one");
numbers.add("two");
numbers.add(0, "three");
numbers.set(2, "four");
numbers.add("five");
numbers.remove(1);
```

Which of the following represents the contents of `numbers` after the code segment has been executed?

(A) `["one", "four", "five"]`

(B) `["three", "two", "five"]`

(C) `["three", "four", "two"]`

(D) `["three", "four", "five"]`

(E) `["one", "two", "three", "four", "five"]`

11. Consider the following method, which is intended to return a list containing the elements of the parameter `myList` with all even elements removed.

```
public static ArrayList<Integer> removeEvens
   (ArrayList<Integer> myList)
{
   for (int i = 0; i < myList.size(); i++)
   {
         if (myList.get(i) % 2 == 0)
         {
             myList.remove(i);
         }
   }
   return myList;
}
```

Which of the following best explains why the code segment does not work as intended?

(A) The code segment causes an `IndexOutOfBoundsException` for all lists because of an incorrect Boolean expression in the for loop.

(B) The code segment causes an `IndexOutOfBoundsException` for lists with at least one even element because the indexes of all subsequent elements change by one when a list element is removed.

(C) The code segment returns a list with fewer elements than intended because it fails to consider the last element of `myList`.

(D) The code segment removes the wrong elements of `myList` because the condition in the `if` statement to test whether an element is even is incorrect.

(E) The code segment skips some elements of `myList` because the indexes of all subsequent elements change by one when a list element is removed.

12. Consider the following code segment.

```
int[][] points = {{11, 12, 13, 14, 15},
                  {21, 22, 23, 24, 25},
                  {31, 32, 33, 34, 35},
                  {41, 42, 43, 44, 45}};
for (int row = 0; row < points.length; row++)
{
   for (int col = points[0].length - 1;
        col >= row; col--)
```

```
    {
        System.out.print(points[row][col] + " ");
    }
    System.out.println();
}
```

What is printed when this code segment is executed?

(A)
```
15 14
   25 24 23
   35 34 33 32
   45 44 43 42 41
```

(B)
```
15 14 13 12
   25 24 23
   35 34
   45
```

(C)
```
11 12 13 14 15
   21 22 23 24
   31 32 33
   41 42
```

(D)
```
15 14 13 12 11
   25 24 23 22
   35 34 33
   45 44
```

(E)
```
15 14 13 12 11
   25 24 23 22 21
   35 34 33 32 31
   45 44 43 42 41
```

13. Consider the following code segment.

```
int[][] arr = {{1,  2,  3,  4},
               {5,  6,  7,  8},
               {9, 10, 11, 12}};
int sum = 0;
for (int[] x : arr)
{
    for (int y = 0; y < x.length - 1; y++)
    {
        sum += x[y];
    }
}
```

What is the value of `sum` as a result of executing the code segment?

(A) 36

(B) 54

(C) 63

(D) 68

(E) 78

14. Consider the following class definitions.

```
public class Thing1
{
   public void calc(int n)
   {
     n *= 3;
     System.out.print(n);
   }
}

public class Thing2 extends Thing1
{
   public void calc(int n)
   {
     n += 2;
     super.calc(n);
     System.out.print(n);
   }
}
```

The following code segment appears in a class other than `Thing1` or `Thing2`.

```
Thing1 t = new Thing2();
t.calc(2);
```

What is printed as a result of executing the code segment?

(A) `4`

(B) `6`

(C) `68`

(D) `124`

(E) `1212`

15. Consider the following two methods, which are intended to return the same values when they are called with the same positive integer parameter `n`.

```
public static int mystery1(int n)
{
   if (n > 1)
   {
     return 5 + mystery1(n - 1);
   }
   else
   {
     return 1;
   }
}
```

```
public static int mystery2(int n)
{
   int total = 0;
   int x = 1;
   while (x < n)
   {
     total += 5;
     x++;
   }
   return total;
}
```

Which, if any, of the following changes to `mystery2` is required so that the two methods work as intended?

(A) The variable `total` should be initialized to `1`.

(B) The variable `x` should be initialized to `0`.

(C) The condition in the `while` loop header should be `x < n - 1`.

(D) The condition in the `while` loop header should be `x <= n`.

(E) No change is required; the methods, as currently written, return the same values when they are called with the same positive integer parameter `n`.

Section II: Free-Response

The following are examples of the kinds of free-response questions found on the exam. Note that on the actual AP Exam, there will be four free-response questions.

Methods and Control Structures (Free-Response Question 1 on the AP Exam)

This question involves the use of *check digits,* which can be used to help detect if an error has occurred when a number is entered or transmitted electronically. An algorithm for computing a check digit, based on the digits of a number, is provided in part (a).

The `CheckDigit` class is shown below. You will write two methods of the `CheckDigit` class.

```
public class CheckDigit
{
    /** Returns the check digit for num, as described in part (a).
     *  Precondition: The number of digits in num is between one and
     *  six, inclusive.
     *            num >= 0
     */
     public static int getCheck(int num)
     {
         /* to be implemented in part (a) */
     }

    /** Returns true if numWithCheckDigit is valid, or false
     *  otherwise, as described in part (b).
     *  Precondition: The number of digits in numWithCheckDigit
     *  is between two and seven, inclusive.
```

```
     *              numWithCheckDigit >= 0
     */
    public static boolean isValid(int numWithCheckDigit)
    {
      /* to be implemented in part (b) */
    }

    /** Returns the number of digits in num. */
    public static int getNumberOfDigits(int num)
    {
      /* implementation not shown */
    }

    /** Returns the nthdigit of num.
     *  Precondition: n >= 1 and n <= the number of digits in num
     */
    public static int getDigit(int num, int n)
    {
      /* implementation not shown */
    }

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

(a) Complete the `getCheck` method, which computes the check digit for a number according to the following rules.

- Multiply the first digit by 7, the second digit (if one exists) by 6, the third digit (if one exists) by 5, and so on. The length of the method's `int` parameter is at most six; therefore, the last digit of a six-digit number will be multiplied by 2.
- Add the products calculated in the previous step.
- Extract the check digit, which is the rightmost digit of the sum calculated in the previous step.

The following are examples of the check-digit calculation.

Example 1, where num has the value 283415

- The sum to calculate is $(2 \times 7) + (8 \times 6) + (3 \times 5) + (4 \times 4) + (1 \times 3) + (5 \times 2) = 14 + 48 + 15 + 16 + 3 + 10 = 106$.
- The check digit is the rightmost digit of 106, or 6, and `getCheck` returns the integer value `6`.

Example 2, where num has the value 2183

- The sum to calculate is $(2 \times 7) + (1 \times 6) + (8 \times 5) + (3 \times 4) = 14 + 6 + 40 + 12 = 72$.
- The check digit is the rightmost digit of 72, or 2, and `getCheck` returns the integer value `2`.

Two helper methods, `getNumberOfDigits` and `getDigit`, have been provided.

- `getNumberOfDigits` returns the number of digits in its `int` parameter.
- `getDigit` returns the nth digit of its `int` parameter.

The following are examples of the use of `getNumberOfDigits` and `getDigit`.

Method Call	Return Value	Explanation
`getNumberOfDigits(283415)`	6	The number `283415` has 6 digits.
`getDigit(283415, 1)`	2	The first digit of `283415` is 2.
`getDigit(283415, 5)`	1	The fifth digit of `283415` is 1.

Complete the `getCheck` method below. You must use `getNumberOfDigits` and `getDigit` appropriately to receive full credit.

```
/** Returns the check digit for num, as described in part (a).
 *  Precondition: The number of digits in num is between one and six,
 *  inclusive.
 *                num >= 0
 */
public static int getCheck(int num)
```

(b) Write the `isValid` method. The method returns `true` if its parameter `numWithCheckDigit`, which represents a number containing a check digit, is valid, and `false` otherwise. The check digit is always the rightmost digit of `numWithCheckDigit`.

The following table shows some examples of the use of `isValid`.

Method Call	Return Value	Explanation
`getCheck(159)`	2	The check digit for `159` is 2.
`isValid(1592)`	`true`	The number `1592` is a valid combination of a number (`159`) and its check digit (`2`).
`isValid(1593)`	`false`	The number `1593` is not a valid combination of a number (`159`) and its check digit (`3`) because `2` is the check digit for `159`.

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

```
/** Returns true if numWithCheckDigit is valid, or false
 *  otherwise, as described in part (b).
 *  Precondition: The number of digits in numWithCheckDigit is
 *  between two and seven, inclusive.
```

```
     *                numWithCheckDigit >= 0
     */
    public static boolean isValid(int numWithCheckDigit)
```

Array/`ArrayList` (Free-Response Question 3 on the AP Exam)

The `Gizmo` class represents `gadgets` that people purchase. Some `Gizmo` objects are electronic and others are not. A partial definition of the `Gizmo` class is shown below.

```
public class Gizmo
{
   /** Returns the name of the manufacturer of this Gizmo. */
   public String getMaker()
   {
      /* implementation not shown */
   }

   /** Returns true if this Gizmo is electronic, and false
    *  otherwise.
    */
   public boolean isElectronic()
   {
      /* implementation not shown */
   }

   /** Returns true if this Gizmo is equivalent to the Gizmo
    *  object represented by the
    *  parameter, and false otherwise.
    */
   public boolean equals(Object other)
   {
      /* implementation not shown */
   }

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

The `OnlinePurchaseManager` class manages a sequence of `Gizmo` objects that an individual has purchased from an online vendor. You will write two methods of the `OnlinePurchaseManager` class. A partial definition of the `OnlinePurchaseManager` class is shown below.

```
public class OnlinePurchaseManager
{
   /** An ArrayList of purchased Gizmo objects,
    *  instantiated in the constructor.
    */
   private ArrayList<Gizmo> purchases;

   /** Returns the number of purchased Gizmo objects that are electronic
    *  whose manufacturer is maker, as described in part (a).
    */
```

```
    public int countElectronicsByMaker(String maker)
    {
        /* to be implemented in part (a) */
    }

    /** Returns true if any pair of adjacent purchased Gizmo objects are
     *  equivalent, and false otherwise, as described in part (b).
     */
    public boolean hasAdjacentEqualPair()
    {
        /* to be implemented in part (b) */
    }

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

(a) Write the `countElectronicsByMaker` method. The method examines the `ArrayList` instance variable `purchases` to determine how many `Gizmo` objects purchased are electronic and are manufactured by `maker`.

Assume that the `OnlinePurchaseManager` object `opm` has been declared and initialized so that the `ArrayList purchases` contains `Gizmo` objects as represented in the following table.

Index in purchases	0	1	2	3	4	5
Value returned by method call `isElectronic()`	`true`	`false`	`true`	`false`	`true`	`false`
Value returned by method call `getMaker()`	`"ABC"`	`"ABC"`	`"XYZ"`	`"lmnop"`	`"ABC"`	`"ABC"`

The following table shows the value returned by some calls to `countElectronicsByMaker`.

Method Call	Return Value
`opm.countElectronicsByMaker("ABC")`	2
`opm.countElectronicsByMaker("lmnop")`	0
`opm.countElectronicsByMaker("XYZ")`	1
`opm.countElectronicsByMaker("QRP")`	0

Complete method `countElectronicsByMaker` below.

```
/** Returns the number of purchased Gizmo objects that are electronic and
 *  whose manufacturer is maker, as described in part (a).
 */
public int countElectronicsByMaker(String maker)
```

(b) When purchasing items online, users occasionally purchase two identical items in rapid succession without intending to do so (e.g., by clicking a purchase button twice). A vendor may want to check a user's purchase history to detect such occurrences and request confirmation.

Write the `hasAdjacentEqualPair` method. The method detects whether two adjacent `Gizmo` objects in `purchases` are equivalent, using the `equals` method of the `Gizmo` class. If an adjacent equivalent pair is found, the `hasAdjacentEqualPair` method returns `true`. If no such pair is found, or if `purchases` has fewer than two elements, the method returns `false`.

Complete method `hasAdjacentEqualPair` below.

```
/** Returns true if any pair of adjacent purchased Gizmo objects are
 *  equivalent, and false otherwise, as described in part (b).
 */
public boolean hasAdjacentEqualPair()
```

Answer Key and Question Alignment to Course Framework

Multiple-Choice Question	Answer	Skill	Learning Objective	Unit
1	C	2.A	CON-1.A	1
2	A	5.A	VAR-1.E.b	2
3	D	1.C	CON-1.D	2
4	E	4.A	CON-2.A	3
5	B	5.A	CON-1.G	3
6	A	2.B	CON-2.C	4
7	C	1.C	MOD-2.D	5
8	A	4.B	VAR-2.B	6
9	A	1.B	VAR-2.C	6
10	D	2.C	VAR-2.D	7
11	E	5.B	VAR-2.E	7
12	D	2.B	VAR-2.G.a	8
13	B	2.B	VAR-2.G.b	8
14	D	2.C	MOD-3.B	9
15	A	4.C	CON-2.O	10

Free-Response Question	Question Type	Skill	Learning Objective	Unit
1	Methods and Control Structures	3.A, 3.C	MOD-1.H, CON-1.A, CON-1.E, CON-2.A CON-2.E	1,2,3,4,5
3	Array / `ArrayList`	3.A, 3.C, 3.D	MOD-1.G, CON-1.B, CON-1.H, CON-2.C, CON-2.J.b, VAR-1.E.b, VAR-2.D, VAR-2.E	1,2,3,4,5,7

The scoring information for the questions within this course and exam description, along with further exam resources, can be found on the **AP Computer Science A Exam Page** on AP Central.

THIS PAGE IS INTENTIONALLY LEFT BLANK.

Scoring Guidelines

Question 1: Methods and Control Structures

This question involves the use of *check digits,* which can be used to help detect if an error has occurred when a number is entered or transmitted electronically. An algorithm for computing a check digit, based on the digits of a number, is provided in part (a).

The `CheckDigit` class is shown below. You will write two methods of the `CheckDigit` class.

```
public class CheckDigit
{
    /** Returns the check digit for num, as described in part (a).
     *  Precondition: The number of digits in num is between one and
     *  six, inclusive.
     *            num >= 0
     */
     public static int getCheck(int num)
     {
         /* to be implemented in part (a) */
     }

    /** Returns true if numWithCheckDigit is valid, or false
     *  otherwise, as described in part (b).
     *  Precondition: The number of digits in numWithCheckDigit
     *  is between two and seven, inclusive.
     *              numWithCheckDigit >= 0
     */
    public static boolean isValid(int numWithCheckDigit)
    {
      /* to be implemented in part (b) */
    }

    /** Returns the number of digits in num. */
   public static int getNumberOfDigits(int num)
    {
      /* implementation not shown */
    }

    /** Returns the nthdigit of num.
     *  Precondition: n >= 1 and n <= the number of digits in num
     */
    public static int getDigit(int num, int n)
    {
      /* implementation not shown */
    }

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

(a) Complete the `getCheck` method, which computes the check digit for a number according to the following rules.

- Multiply the first digit by 7, the second digit (if one exists) by 6, the third digit (if one exists) by 5, and so on. The length of the method's `int` parameter is at most six; therefore, the last digit of a six-digit number will be multiplied by 2.
- Add the products calculated in the previous step.
- Extract the check digit, which is the rightmost digit of the sum calculated in the previous step.

The following are examples of the check-digit calculation.

Example 1, where num has the value 283415

- The sum to calculate is $(2 \times 7) + (8 \times 6) + (3 \times 5) + (4 \times 4) + (1 \times 3) + (5 \times 2) = 14 + 48 + 15 + 16 + 3 + 10 = 106$.
- The check digit is the rightmost digit of 106, or 6, and `getCheck` returns the integer value `6`.

Example 2, where num has the value 2183

- The sum to calculate is $(2 \times 7) + (1 \times 6) + (8 \times 5) + (3 \times 4) = 14 + 6 + 40 + 12 = 72$.
- The check digit is the rightmost digit of 72, or 2, and `getCheck` returns the integer value `2`.

Two helper methods, `getNumberOfDigits` and `getDigit`, have been provided.

- `getNumberOfDigits` returns the number of digits in its `int` parameter.
- `getDigit` returns the nth digit of its `int` parameter.

The following are examples of the use of `getNumberOfDigits` and `getDigit`.

Method Call	Return Value	Explanation
`getNumberOfDigits(283415)`	6	The number `283415` has 6 digits.
`getDigit(283415, 1)`	2	The first digit of `283415` is `2`.
`getDigit(283415, 5)`	1	The fifth digit of `283415` is `1`.

Complete the `getCheck` method below. You must use `getNumberOfDigits` and `getDigit` appropriately to receive full credit.

```
/** Returns the check digit for num, as described in part (a).
 *  Precondition: The number of digits in num is between one and six,
 *  inclusive.
 *              num >= 0
 */
public static int getCheck(int num)
```

(b) Write the `isValid` method. The method returns `true` if its parameter `numWithCheckDigit`, which represents a number containing a check digit, is valid, and `false` otherwise. The check digit is always the rightmost digit of `numWithCheckDigit`.

The following table shows some examples of the use of `isValid`.

Method Call	Return Value	Explanation
`getCheck(159)`	`2`	The check digit for `159` is `2`.
`isValid(1592)`	`true`	The number `1592` is a valid combination of a number (`159`) and its check digit (`2`).
`isValid(1593)`	`false`	The number `1593` is not a valid combination of a number (`159`) and its check digit (`3`) because `2` is the check digit for `159`.

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

```
/** Returns true if numWithCheckDigit is valid, or false
 *  otherwise, as described in part (b).
 *  Precondition: The number of digits in numWithCheckDigit is
 *  between two and seven, inclusive.
 *                  numWithCheckDigit >= 0
 */
public static boolean isValid(int numWithCheckDigit)
```

Applying the Scoring Criteria

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

1-Point Penalty

v) Array/collection access confusion `([] get)`

w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)

x) Local variables used but none declared

y) Destruction of persistent data (e.g., changing value referenced by parameter)

z) Void method or constructor that returns a value

No Penalty

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

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

Scoring Guidelines for Question 1: Methods and Control Structures — 9 points

Learning Objectives: MOD-1.H CON-1.A CON-1.E CON-2.A CON-2.E

Canonical solution

(a) — 4 points

```
public static int getCheck(int num)
{
    int sum = 0;
    for (int i = 1; i <= getNumberOfDigits(num); i++)
    {
      sum += (8 - i) * getDigit(num, i);
    }
    return sum % 10;
}
```

(b) — 5 points

```
public static boolean isValid(int numWithCheckDigit)
{
    int check = numWithCheckDigit % 10;
    int num = numWithCheckDigit / 10;
    int newCheck = getCheck(num);
    if (check == newCheck)
    {
      return true;
    }
    else
    {
      return false;
    }
}
```

(a) `getCheck`

	Scoring Criteria	Decision Rules	
1	Calls `getNumberOfDigits` and `getDigit`		**1 point** 3.A MOD-1.H
2	Calculates one partial sum	**Responses earn the point if they...** • Calculate the sum inside or outside the context of a loop.	**1 point** 3.C CON-1.A
3	Calculates all partial sums (*in the context of a loop, no bounds errors*)	**Responses earn the point if they...** • Use either a `for` or `while` loop.	**1 point** 3.C CON-2.E
4	Returns the last digit of the calculated sum as the check digit	**Responses still earn the point even if they...** • Calculate the sum incorrectly.	**1 point** 3.C CON-1.A
		Total for part (a)	**4 points**

(b) `isValid`

	Scoring Criteria	Decision Rules	
5	Obtains check digit of `numWithCheckDigit`		**1 point** 3.C CON-1.A
6	Obtains number remaining in `numWithCheckDigit` after check digit removed		**1 point** 3.C CON-1.A
7	Calls `getCheck` on number without check digit		**1 point** 3.A MOD-1.H
8	Compares check digit of `numWithCheckDigit` and return value from `getCheck`	**Responses still earn the point even if they...** • Calculated check digit incorrectly; or • Calculated number without check digit incorrectly.	**1 point** 3.C CON-1.E
9	Returns `true` or `false` depending on the result of the previous comparison	**Responses still earn the point even if they...** • Do not use an if statement	**1 point** 3.C CON-2.A
		Total for part (b)	**5 points**

Question specific penalties

None

Total for question 1 **9 points**

Question 3: Array/ArrayList

The Gizmo class represents gadgets that people purchase. Some Gizmo objects are electronic and others are not. A partial definition of the Gizmo class is shown below.

```
public class Gizmo
{
   /** Returns the name of the manufacturer of this Gizmo. */
   public String getMaker()
   {
      /* implementation not shown */
   }

   /** Returns true if this Gizmo is electronic, and false
    *  otherwise.
    */
   public boolean isElectronic()
   {
      /* implementation not shown */
   }

   /** Returns true if this Gizmo is equivalent to the Gizmo
    *  object represented by the
    *  parameter, and false otherwise.
    */
   public boolean equals(Object other)
   {
      /* implementation not shown */
   }

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

The OnlinePurchaseManager class manages a sequence of Gizmo objects that an individual has purchased from an online vendor. You will write two methods of the OnlinePurchaseManager class. A partial definition of the OnlinePurchaseManager class is shown below.

```
public class OnlinePurchaseManager
{
   /** An ArrayList of purchased Gizmo objects,
    *  instantiated in the constructor.
    */
   private ArrayList<Gizmo> purchases;

   /** Returns the number of purchased Gizmo objects that are electronic
    *  whose manufacturer is maker, as described in part (a).
    */
   public int countElectronicsByMaker(String maker)
   {
      /* to be implemented in part (a) */
   }
```

```
    /** Returns true if any pair of adjacent purchased Gizmo objects are
     *  equivalent, and false otherwise, as described in part (b).
     */
    public boolean hasAdjacentEqualPair()
    {
        /* to be implemented in part (b) */
    }

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

(a) Write the `countElectronicsByMaker` method. The method examines the `ArrayList` instance variable `purchases` to determine how many `Gizmo` objects purchased are electronic and are manufactured by `maker`.

Assume that the `OnlinePurchaseManager` object `opm` has been declared and initialized so that the `ArrayList purchases` contains `Gizmo` objects as represented in the following table.

Index in purchases	0	1	2	3	4	5
Value returned by method call `isElectronic()`	`true`	`false`	`true`	`false`	`true`	`false`
Value returned by method call `getMaker()`	`"ABC"`	`"ABC"`	`"XYZ"`	`"lmnop"`	`"ABC"`	`"ABC"`

The following table shows the value returned by some calls to `countElectronicsByMaker`.

Method Call	Return Value
`opm.countElectronicsByMaker("ABC")`	2
`opm.countElectronicsByMaker("lmnop")`	0
`opm.countElectronicsByMaker("XYZ")`	1
`opm.countElectronicsByMaker("QRP")`	0

Complete method `countElectronicsByMaker` below.

```
/** Returns the number of purchased Gizmo objects that are electronic and
 *  whose manufacturer is maker, as described in part (a).
 */
public int countElectronicsByMaker(String maker)
```

(b) When purchasing items online, users occasionally purchase two identical items in rapid succession without intending to do so (e.g., by clicking a purchase button twice). A vendor may want to check a user's purchase history to detect such occurrences and request confirmation.

Write the `hasAdjacentEqualPair` method. The method detects whether two adjacent `Gizmo` objects in `purchases` are equivalent, using the `equals` method of the `Gizmo` class. If an adjacent equivalent pair is found, the `hasAdjacentEqualPair` method returns `true`. If no such pair is found, or if `purchases` has fewer than two elements, the method returns `false`.

Complete method `hasAdjacentEqualPair` below.

```
/** Returns true if any pair of adjacent purchased Gizmo objects are
 *  equivalent, and false otherwise, as described in part (b).
 */
public boolean hasAdjacentEqualPair()
```

Scoring Guidelines for Question 3: Array/`ArrayList` — 9 points

Learning Objectives: MOD-1.G | CON-1.B | CON-1.H | CON-2.C | CON-2.J.B | VAR-1.E.B | VAR-2.D | VAR 2.E

Canonical solution

(a) — 4 points

```
public int countElectronicsByMarker(String maker)
{
    int result = 0;
    for (Gizmo g : purchases)
    {
      if (g.getMaker().equals(maker) && g.isElectronic())
      {
          result++;
      }
    }
    return result;
}
```

(b) — 5 points

```
public boolean hasAdjacentEqualPair()
{
    Gizmo g1 = purchases.get(0);
    for (int pos = 1; pos < purchases.size(); pos++)
    {
      Gizmo g2 = purchases.get(pos);
      if (g1 . equals(g2))
      {
          return true;
      }
      g1 = g2;
    }
    return false;
}
```

(a) `countElectronicsByMaker`

	Scoring Criteria	Decision Rules	
1	Accesses all elements of `purchases` (*no bounds errors*)	**Responses earn the point if they...** • Use a for, enhanced for, or while loop.	**1 point** 3.D VAR 2.E
2	Calls `isElectronic` to test whether or not a `Gizmo` object is electronic and `getMaker` to get the name of the manufacturer of a `Gizmo` object	**Responses earn the point if they...** • Call methods on any Gizmo object, regardless of whether this is in the context of a loop.	**1 point** 3.A MOD-1.G
3	Tests whether or not the maker of a `Gizmo` object matches `maker`, using an appropriate `String` comparison at least once, in the context of a loop		**1 point** 3.C VAR-1.E.b
4	Computes the number of elements that are electronic and made by `maker`	**Responses still earn the point even if they...** • Do not return the number of elements that are electronic and made by `maker`	**1 point** 3.C CON-1.B
		Total for part (a)	**4 points**

(b) `hasAdjacentEqualPair`

	Scoring Criteria	Decision Rules	
5	Determines whether the `ArrayList` contains at least two elements		**1 point** 3.D VAR-2.D
6	Accesses all necessary elements of `purchases` (*no bounds error*)		**1 point** 3.D CON-2.J.b
7	Accesses two adjacent elements of the `ArrayList` in the context of a comparison	**Responses still earn the point even if they...** • Do not access adjacent elements in the context of a loop.	**1 point** 3.D VAR 2.E
8	Compares two distinct elements of the `ArrayList` for equivalence		**1 point** 3.C CON-1.H
9	Returns `true` if at least one equivalent pair is found, and `false` otherwise or if the `ArrayList` has fewer than two elements	**Responses earn the point if they...** • Use a `for` or `while` loop to access the pairs of elements.	**1 point** 3.C CON-2.C
		Total for part (b)	**5 points**

Question specific penalties

None

Total for question 3 **9 points**