AP® Computer Science A Exam

SECTION I: Multiple Choice

DO NOT OPEN THIS BOOKLET UNTIL YOU ARE TOLD TO DO SO.

At a Glance

Total Time
1 hour and 30 minutes
Number of Questions
40
Percent of Total Score
50%
Writing Instrument
Pencil required
Electronic Device
None allowed

Instructions

The Java Quick Reference is located inside the front cover of this booklet.

Section I of this exam contains 40 multiple-choice questions.

Indicate all of your answers to the multiple-choice questions on the answer sheet. No credit will be given for anything written in this exam booklet, but you may use the booklet for notes or scratch work.

Use your time effectively, working as quickly as you can without losing accuracy. Do not spend too much time on any one question. Go on to other questions and come back to the ones you have not answered if you have time. It is not expected that everyone will know the answers to all of the multiple-choice questions.

Your total score on the multiple-choice section is based only on the number of questions answered correctly. Points are not deducted for incorrect answers or unanswered questions.

Java Quick Reference

Accessible methods from the Java library that may be included in the exam

Class Constructors and Methods	Explanation
String Class	
`String(String str)`	Constructs a new `String` object that represents the same sequence of characters as `str`
`int length()`	Returns the number of characters in a `String` object
`String substring(int from, int to)`	Returns the substring beginning at index `from` and ending at index `to - 1`
`String substring(int from)`	Returns `substring(from, length())`
`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found
`boolean equals(String other)`	Returns `true` if `this` is equal to `other`; returns `false` otherwise
`int compareTo(String other)`	Returns a value `<0` if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value `>0` if `this` is greater than `other`
Integer Class	
`Integer(int value)`	Constructs a new `Integer` object that represents the specified `int` value
`Integer.MIN_VALUE`	The minimum value represented by an `int` or `Integer`
`Integer.MAX_VALUE`	The maximum value represented by an `int` or `Integer`
`int intValue()`	Returns the value of this `Integer` as an `int`
Double Class	
`Double(double value)`	Constructs a new `Double` object that represents the specified `double` value
`double doubleValue()`	Returns the value of this `Double` as a `double`
Math Class	
`static int abs(int x)`	Returns the absolute value of an `int` value
`static double abs(double x)`	Returns the absolute value of a `double` value
`static double pow(double base, double exponent)`	Returns the value of the first parameter raised to the power of the second parameter
`static double sqrt(double x)`	Returns the positive square root of a `double` value
`static double random()`	Returns a `double` value greater than or equal to `0.0` and less than `1.0`
ArrayList Class	
`int size()`	Returns the number of elements in the list
`boolean add(E obj)`	Appends `obj` to end of list; returns `true`
`void add(int index, E obj)`	Inserts `obj` at position `index (0 <= index <= size)`, moving elements at position `index` and higher to the right (adds 1 to their indices) and adds 1 to size
`E get(int index)`	Returns the element at position `index` in the list
`E set(int index, E obj)`	Replaces the element at position `index` with `obj`; returns the element formerly at position `index`
`E remove(int index)`	Removes element from position `index`, moving elements at position `index + 1` and higher to the left (subtracts 1 from their indices) and subtracts 1 from size; returns the element formerly at position `index`
Object Class	
`boolean equals(Object other)`	
`String toString()`	

NO TEST MATERIAL ON THIS PAGE

COMPUTER SCIENCE A

SECTION I

Time—1 hour and 30 minutes

40 Questions

Directions: Determine the answer to each of the following questions or incomplete statements, using the available space for any necessary scratch work. Then decide which is the best of the choices given and then enter the letter in the corresponding space on the answer sheet. No credit will be given for anything written in the exam booklet. Do not spend too much time on any one problem.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Assume that declarations of variables and methods appear within the context of an enclosing class.
- Assume that method calls that are not prefixed with an object or class name and are not shown within a complete class definition appear within the context of an enclosing class.
- 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.

1. Consider the following code segment.

```
int a = 3 + 2 * 3;
int b = 4 + 3 / 2;
int c = 7 % 4 + 3;
double d = a + b + c;
```

What is the value of `d` after the code segment is executed?

(A) `14.0`

(B) `18.0`

(C) `20.0`

(D) `20.5`

(E) `26.0`

---

2. Consider the following code segment. Assume `num` is a properly declared and initialized `int` variable.

```
if (num > 0)
{
   if (num % 2 == 0)
   {
      System.out.println("A");
   }
   else
   {
      System.out.println("B");
   }
}
```

Which of the following best describes the result of executing the code segment?

(A) When `num` is a negative odd integer, `"B"` is printed; otherwise, `"A"` is printed.

(B) When `num` is a negative even integer, `"B"` is printed; otherwise, nothing is printed.

(C) When `num` is a positive even integer, `"A"` is printed; otherwise, `"B"` is printed.

(D) When `num` is a positive even integer, `"A"` is printed; when `num` is a positive odd integer, `"B"` is printed; otherwise, nothing is printed.

(E) When `num` is a positive odd integer, `"A"` is printed; when `num` is a positive even integer, `"B"` is printed; otherwise, nothing is printed.

3. Consider the method `getHours`, which is intended to calculate the number of hours that a vehicle takes to travel between two *mile markers* on a highway if the vehicle travels at a constant speed of 60 miles per hour. A mile marker is a sign showing the number of miles along a road between some fixed location (for example, the beginning of a highway) and the current location.

   The following table shows two examples of the intended behavior of `getHours`, based on the `int` parameters `marker1` and `marker2`.

marker1	marker2	**Return Value**
100	220	2.0
100	70	0.5

   Consider the following implementation of `getHours`.

```
public static double getHours(int marker1, int marker2)
{
   /* missing statement */
   return hours;
}
```

   Which of the following statements can replace `/* missing statement */` so `getHours` works as intended?

   (A) `double hours = (Math.abs(marker1) - Math.abs(marker2)) / 60.0;`

   (B) `double hours = Math.abs(marker1 - marker2 / 60.0);`

   (C) `double hours = Math.abs(marker1 - marker2) / 60.0;`

   (D) `double hours = Math.abs((marker1 - marker2) / 60);`

   (E) `double hours = (double) (Math.abs(marker1 - marker2) / 60);`

4. Consider the following method.

```
public static void message(int a, int b, int c)
{
   if (a < 10)
   {
      if (b < 10)
      {
         System.out.print("X");
      }
      System.out.print("Y");
   }
   if (c < 10)
   {
      if (b > 10)
      {
         System.out.print("Y");
      }
      else
      {
         System.out.print("Z");
      }
   }
}
```

What is printed as a result of the call `message(5, 15, 5)` ?

(A) `XY`

(B) `XYZ`

(C) `Y`

(D) `YY`

(E) `Z`

5. Consider the following class definition.

```
public class Bird
{
   private String species;
   private String color;
   private boolean canFly;

   public Bird(String str, String col, boolean cf)
   {
      species = str;
      color = col;
      canFly = cf;
   }
}
```

Which of the following constructors, if added to the `Bird` class, will cause a compilation error?

(A)
```
public Bird()
{
   species = "unknown";
   color = "unknown";
   canFly = false;
}
```

(B)
```
public Bird(boolean cf)
{
   species = "unknown";
   color = "unknown";
   canFly = cf;
}
```

(C)
```
public Bird(String col, String str)
{
   species = str;
   color = col;
   canFly = false;
}
```

(D)
```
public Bird(boolean cf, String str, String col)
{
   species = str;
   color = col;
   canFly = cf;
}
```

(E)
```
public Bird(String col, String str, boolean cf)
{
   species = str;
   color = col;
   canFly = cf;
}
```

---

6. Which of the following expressions evaluate to `3.5` ?

   I. `(double) 2 / 4 + 3`

   II. `(double) (2 / 4) + 3`

   III. `(double) (2 / 4 + 3)`

   (A) I only

   (B) III only

   (C) I and II only

   (D) II and III only

   (E) I, II, and III

7. Consider the following code segment.

```
int num = /* initial value not shown */;
boolean b1 = true;
if (num > 0)
{
   if (num >= 100)
   {
      b1 = false;
   }
}
else
{
   if (num >= -100)
   {
      b1 = false;
   }
}
```

Which of the following statements assigns the same value to `b2` as the code segment assigns to `b1` for all values of `num` ?

(A) `boolean b2 = (num > -100) && (num < 100);`

(B) `boolean b2 = (num > -100) || (num < 100);`

(C) `boolean b2 = (num < -100) || (num > 100);`

(D) `boolean b2 = (num < -100) && (num > 0 || num < 100);`

(E) `boolean b2 = (num < -100) || (num > 0 && num < 100);`

8. Consider the following class definition.

```
public class Points
{
   private double num1;
   private double num2;

   public Points(int n1, int n2)            // Line 6
   {
      num1 = n1;                            // Line 8
      num2 = n2;                            // Line 9
   }

   public void incrementPoints(int value)   // Line 12
   {
      n1 += value;                          // Line 14
      n2 += value;                          // Line 15
   }
}
```

The class does not compile. Which of the following identifies the error in the class definition?

(A) In line 6, the `Points` constructor must have a `void` return type.

(B) In lines 8 and 9, `int` values cannot be assigned to `double` variables.

(C) In line 12, the `incrementPoints` method must have a non-`void` return type.

(D) In lines 14 and 15, the variables `n1` and `n2` are not defined.

(E) In lines 14 and 15, the variable `value` is not defined.

9. Consider the following code segment.

```
ArrayList<Integer> numList = new ArrayList<Integer>();

numList.add(3);
numList.add(2);
numList.add(1);
numList.add(1, 0);
numList.set(0, 2);

System.out.print(numList);
```

What is printed by the code segment?

(A) `[1, 3, 0, 1]`

(B) `[2, 0, 2, 1]`

(C) `[2, 0, 2, 3]`

(D) `[2, 3, 2, 1]`

(E) `[3, 0, 0, 1]`

---

10. Consider the following method.

```
public static void printSome(int num1, int num2)
{
   for (int i = 0; i < num1; i++)
   {
      if (i % num2 == 0 && i % 2 == 0)
      {
         System.out.print(i + " ");
      }
   }
}
```

Which of the following method calls will cause `"0 10 "` to be printed?

(A) `printSome(0, 20)`

(B) `printSome(5, 10)`

(C) `printSome(10, 5)`

(D) `printSome(20, 5)`

(E) `printSome(25, 5)`

11. Which of the following code segments produces the output `"987654321"` ?

(A)
```
int num = 10;
while (num > 0)
{
   System.out.print(num);
   num--;
}
```

(B)
```
int num = 10;
while (num >= 0)
{
   System.out.print(num);
   num--;
}
```

(C)
```
int num = 10;
while (num > 1)
{
   num--;
   System.out.print(num);
}
```

(D)
```
int num = 10;
while (num >= 1)
{
   num--;
   System.out.print(num);
}
```

(E)
```
int num = 0;
while (num <= 9)
{
   System.out.print(10 - num);
   num++;
}
```

12. Consider the following class definitions.

```
public class Person
{
   private String name;

   public String getName()
   {  return name;  }
}

public class Book
{
   private String author;
   private String title;
   private Person borrower;

   public Book(String a, String t)
   {
      author = a;
      title = t;
      borrower = null;
   }

   public void printDetails()
   {
      System.out.print("Author: " + author + " Title: " + title);

      if ( /* missing condition */ )
      {
         System.out.println(" Borrower: " + borrower.getName());
      }
   }

   public void setBorrower(Person b)
   {  borrower = b;  }
}
```

Which of the following can replace `/* missing condition */` so that the `printDetails` method CANNOT cause a run-time error?

I. `!borrower.equals(null)`

II. `borrower != null`

III. `borrower.getName() != null`

(A) I only

(B) II only

(C) III only

(D) I and II

(E) II and III

---

13. Assume that `a`, `b`, and `c` are `boolean` variables that have been properly declared and initialized. Which of the following `boolean` expressions is equivalent to `!(a && b) || c` ?

(A) `a && b && c`

(B) `a || b || c`

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

(D) `!a && !b && c`

(E) `!a || !b || c`

14. The following categories are used by some researchers to categorize zip codes as urban, suburban, or rural based on population density.

    - An urban zip code is a zip code with more than 3,000 people per square mile.
    - A suburban zip code is a zip code with between 1,000 and 3,000 people, inclusive, per square mile.
    - A rural zip code is a zip code with fewer than 1,000 people per square mile.

    Consider the following method, which is intended to categorize a zip code as urban, suburban, or rural based on the population density of the area included in the zip code.

```
public static String getCategory(int density)
{
   /* missing code */
}
```

    Which of the following code segments can replace `/* missing code */` so the `getCategory` method works as intended?

I.
```
String cat;
if (density > 3000)
{
   cat = "urban";
}
else if (density > 999)
{
   cat = "suburban";
}
else
{
   cat = "rural";
}
return cat;
```

II.
```
String cat;
if (density > 3000)
{
   cat = "urban";
}
if (density > 999)
{
   cat = "suburban";
}
cat = "rural";
return cat;
```

III.
```
if (density > 3000)
{
   return "urban";
}
if (density > 999)
{
   return "suburban";
}
return "rural";
```

(A) I only

(B) III only

(C) I and II only

(D) I and III only

(E) I, II, and III

---

15. Consider the following code segment. Assume that `a` is greater than zero.

```
int a = /* value not shown */;
int b = a + (int) (Math.random() * a);
```

Which of the following best describes the value assigned to `b` when the code segment is executed?

(A) `a`

(B) `2 * a`

(C) A random integer between `0` and `a - 1`, inclusive

(D) A random integer between `a` and `2 * a`, inclusive

(E) A random integer between `a` and `2 * a - 1`, inclusive

16. Consider the following recursive method.

```
public static void stars(int num)
{
   if (num == 1)
   {
      return;
   }

   stars(num - 1);

   for (int i = 0; i < num; i++)
   {
      System.out.print("*");
   }
   System.out.println();
}
```

What is printed as a result of the method call `stars(5)` ?

(A)
```
*****
```

(B)
```
**
***
****
*****
```

(C)
```
*
**
***
****
*****
```

(D)
```
*****
****
***
**
```

(E)
```
*****
****
***
**
*
```

17. Consider the following class definitions.

```
public class Hero
{
   private String name;
   private int power;

   public Hero(String n, int p)
   {
      name = n;
      power = p;
   }

   public void powerUp(int p)
   {
      power += p;
   }

   public int showPower()
   {   return power;   }
}

public class SuperHero extends Hero
{
   public SuperHero(String n, int p)
   {
      super(n, p);
   }

   public void powerUp(int p)
   {
      super.powerUp(p * 2);
   }
}
```

The following code segment appears in a class other than `Hero` and `SuperHero`.

```
Hero j = new SuperHero("JavaHero", 50);
j.powerUp(10);
System.out.println(j.showPower());
```

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

(A) `10`

(B) `20`

(C) `60`

(D) `70`

(E) `100`

18. Consider the following method, which is intended to return the number of *local maximum* values in an array. Local maximum values are array elements that are greater than both adjacent array elements. The first and last elements of an array have only a single adjacent element, so neither the first nor the last array element is counted by this method. For example, an array containing the values `{3, 9, 7, 4, 10, 12, 3, 8}` has two local maximum values: `9` and `12`.

```
public static int countPeaks(int[] data)
{
   int numPeaks = 0;

   for ( /* missing loop header */ )
   {
      if (data[p - 1] < data[p] && data[p] > data[p + 1])
      {
         numPeaks++;
      }
   }
   return numPeaks;
}
```

Which of the following can replace `/* missing loop header */` so the method `countPeaks` works as intended?

(A) `int p = data.length - 1; p > 0; p--`

(B) `int p = 0; p < data.length; p++`

(C) `int p = 0; p < data.length - 1; p++`

(D) `int p = 1; p < data.length; p++`

(E) `int p = 1; p < data.length - 1; p++`

19. Consider the following code segment.

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

for (int j = 0; j < values.length; j++)
{
   for (int k = 0; k < values[0].length; k++)
   {
      if (k == 0)
      {
         values[j][k] *= 2;
      }
      x += values[j][k];
   }
}
```

What is the value of `x` after the code segment is executed?

(A) 7

(B) 17

(C) 21

(D) 26

(E) 27

20. Consider the following class definition.

```
public class Book
{
   private int pages;

   public int getPages()
   {
      return pages;
   }

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

The following code segment is intended to store in `maxPages` the greatest number of pages found in any `Book` object in the array `bookArr`.

```
Book[] bookArr = { /* initial values not shown */ };
int maxPages = bookArr[0].getPages();

for (Book b : bookArr)
{
   /* missing code */
}
```

Which of the following can replace `/* missing code */` so the code segment works as intended?

(A)
```
if (b.pages > maxPages)
{
   maxPages = b.pages;
}
```

(B)
```
if (b.getPages() > maxPages)
{
   maxPages = b.getPages();
}
```

(C)
```
if (Book[b].pages > maxPages)
{
   maxPages = Book[b].pages;
}
```

(D)
```
if (bookArr[b].pages > maxPages)
{
   maxPages = bookArr[b].pages;
}
```

(E)
```
if (bookArr[b].getPages() > maxPages)
{
   maxPages = bookArr[b].getPages();
}
```

Questions 21 - 22 refer to the information below.

Consider the following method.

```
public static String[] strArrMethod(String[] arr)
{
   String[] result = new String[arr.length];

   for (int j = 0; j < arr.length; j++)
   {
      String sm = arr[j];
      for (int k = j + 1; k < arr.length; k++)
      {
         if (arr[k].length() < sm.length())
         {
            sm = arr[k];    // Line 12
         }
      }
      result[j] = sm;
   }
   return result;
}
```

21. Consider the following code segment.

```
String[] testOne = {"first", "day", "of", "spring"};
String[] resultOne = strArrMethod(testOne);
```

What are the contents of `resultOne` when the code segment has been executed?

(A) `{"day", "first", "of", "spring"}`

(B) `{"of", "day", "first", "spring"}`

(C) `{"of", "day", "of", "spring"}`

(D) `{"of", "of", "of", "spring"}`

(E) `{"spring", "first", "day", "of"}`

22. Consider the following code segment.

```
String[] testTwo = {"last", "day", "of", "the", "school", "year"};
String[] resultTwo = strArrMethod(testTwo);
```

How many times is the line labeled `// Line 12` in the `strArrMethod` executed as a result of executing the code segment?

(A) 4 times

(B) 5 times

(C) 6 times

(D) 15 times

(E) 30 times

23. Consider the following method, which is intended to print the values in its two-dimensional integer array parameter in row-major order.

```
public static void rowMajor(int[][] arr)
{
   /* missing code */
}
```

As an example, consider the following code segment.

```
int[][] theArray = {{1, 2}, {3, 4}, {5, 6}, {7, 8}};
rowMajor(theArray);
```

When executed, the code segment should produce the following output.

```
1 2 3 4 5 6 7 8
```

Which of the following code segments can replace `/* missing code */` so that the `rowMajor` method works as intended?

(A)
```
for (int j : arr)
{
   for (int k : j)
   {
      System.out.print(j + " ");
   }
}
```

(B)
```
for (int j : arr)
{
   for (int k : j)
   {
      System.out.print(k + " ");
   }
}
```

(C)
```
for (int[] j : arr)
{
   for (int k : j)
   {
      System.out.print(j + " ");
   }
}
```

(D)
```
for (int[] j : arr)
{
   for (int k : j)
   {
      System.out.print(k + " ");
   }
}
```

(E)
```
for (int[] j : arr)
{
   for (int k : j)
   {
      System.out.print(arr[k] + " ");
   }
}
```

24. Consider the following class definition.

```
public class SomeClass
{
   private int x = 0;
   private static int y = 0;

   public SomeClass(int pX)
   {
      x = pX;
      y++;
   }

   public void incrementY()
   {   y++;  }

   public void incrementY(int inc)
   {   y += inc;  }

   public int getY()
   {   return y;  }
}
```

The following code segment appears in a class other than `SomeClass`.

```
SomeClass first = new SomeClass(10);
SomeClass second = new SomeClass(20);
SomeClass third = new SomeClass(30);
first.incrementY();
second.incrementY(10);
System.out.println(third.getY());
```

What is printed as a result of executing the code segment if the code segment is the first use of a `SomeClass` object?

(A) `0`

(B) `1`

(C) `11`

(D) `14`

(E) `30`

25. Consider the following method.

```
public static String rearrange(String str)
{
   String temp = "";

   for (int i = str.length() - 1; i > 0; i--)
   {
      temp += str.substring(i - 1, i);
   }

   return temp;
}
```

What, if anything, is returned by the method call `rearrange("apple")` ?

(A) `"appl"`

(B) `"apple"`

(C) `"elppa"`

(D) `"lppa"`

(E) Nothing is returned due to a run-time error.

26. Consider the following two code segments. Assume that the `int` variables `m` and `n` have been properly declared and initialized and are both greater than `0`.

I.
```
for (int i = 0; i < m * n; i++)
{
   System.out.print("A");
}
```

II.
```
for (int j = 1; j <= m; j++)
{
   for (int k = 1; k < n; k++)
   {
      System.out.print("B");
   }
}
```

Assume that the initial values of `m` and `n` are the same in code segment I as they are in code segment II. Which of the following correctly compares the number of times that `"A"` and `"B"` are printed when each code segment is executed?

(A) `"A"` is printed `m` fewer times than `"B"`.

(B) `"A"` is printed `n` fewer times than `"B"`.

(C) `"A"` is printed `m` more times than `"B"`.

(D) `"A"` is printed `n` more times than `"B"`.

(E) `"A"` and `"B"` are printed the same number of times.

27. Consider the following statement. Assume that `a` and `b` are properly declared and initialized `boolean` variables.

```
boolean c = (a && b) || (!a && b);
```

Under which of the following conditions will `c` be assigned the value `false` ?

(A) Always

(B) Never

(C) When `a` and `b` have the same value

(D) When `a` has the value `false`

(E) When `b` has the value `false`

---

28. Consider the following method.

```
public static String abMethod(String a, String b)
{
   int x = a.indexOf(b);

   while (x >= 0)
   {
      a = a.substring(0, x) + a.substring(x + b.length());
      x = a.indexOf(b);
   }

   return a;
}
```

What, if anything, is returned by the method call `abMethod("sing the song", "ng")` ?

(A) `"si"`

(B) `"si the so"`

(C) `"si the song"`

(D) `"sig the sog"`

(E) Nothing is returned because a `StringIndexOutOfBoundsException` is thrown.

29. Consider the following method.

```
public static int calcMethod(int num)
{
   if (num == 0)
   {
      return 10;
   }
   return num + calcMethod(num / 2);
}
```

What value is returned by the method call `calcMethod(16)` ?

(A) `10`

(B) `26`

(C) `31`

(D) `38`

(E) `41`

30. Consider the following class definitions.

```
public class Rectangle
{
   private int height;
   private int width;

   public Rectangle()
   {
      height = 1;
      width = 1;
   }

   public Rectangle(int x)
   {
      height = x;
      width = x;
   }

   public Rectangle(int h, int w)
   {
      height = h;
      width = w;
   }

   // There may be methods that are not shown.
}

public class Square extends Rectangle
{
   public Square(int x)
   {
      /* missing code */
   }
}
```

Which of the following code segments can replace `/* missing code */` so that the `Square` class constructor initializes the `Rectangle` class instance variables `height` and `width` to `x` ?

(A)
```
super();
```

(B)
```
super(x);
```

(C)
```
Rectangle(x);
```

(D)
```
Square(x, x);
```

(E)
```
height = x;
width = x;
```

31. Consider an integer array `nums`, which has been properly declared and initialized with one or more values. Which of the following code segments counts the number of negative values found in `nums` and stores the count in `counter` ?

I.
```
int counter = 0;
int i = -1;
while (i <= nums.length - 2)
{
   i++;
   if (nums[i] < 0)
   {
      counter++;
   }
}
```

II.
```
int counter = 0;
for (int i = 1; i < nums.length; i++)
{
   if (nums[i] < 0)
   {
      counter++;
   }
}
```

III.
```
int counter = 0;
for (int i : nums)
{
   if (nums[i] < 0)
   {
      counter++;
   }
}
```

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

32. Consider the following class definitions.

```
public class ClassA
{
   public String getValue()
   {
      return "A";
   }

   public void showValue()
   {
      System.out.print(getValue());
   }
}

public class ClassB extends ClassA
{
   public String getValue()
   {
      return "B";
   }
}
```

The following code segment appears in a class other than `ClassA` or `ClassB`.

```
ClassA obj = new ClassB();
obj.showValue();
```

What, if anything, is printed when the code segment is executed?

(A) `A`

(B) `B`

(C) `AB`

(D) `BA`

(E) Nothing is printed because the code does not compile.

33. Consider the following code segment.

```
String[][] letters = {{"A", "B", "C", "D"},
                      {"E", "F", "G", "H"},
                      {"I", "J", "K", "L"}};

for (int col = 1; col < letters[0].length; col++)
{
   for (int row = 1; row < letters.length; row++)
   {
      System.out.print(letters[row][col] + " ");
   }

   System.out.println();
}
```

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

(A)
```
A E I
F J
K
```

(B)
```
B F J
C G K
D H L
```

(C)
```
E I
F J
G K
H L
```

(D)
```
F G H
J K L
```

(E)
```
F J
G K
H L
```

34. The following method is intended to remove all elements of an `ArrayList` of integers that are divisible by `key` and add the removed elements to a new `ArrayList`, which the method returns.

```
public static ArrayList<Integer> match(ArrayList<Integer> numList, int key)
{
   ArrayList<Integer> returnList = new ArrayList<Integer>();

   int i = 0;
   while (i < numList.size())
   {
      int num = numList.get(i);
      if (num % key == 0)
      {
         numList.remove(i);
         returnList.add(num);
      }
      i++;
   }
   return returnList;
}
```

As an example, if the method is called with an `ArrayList` containing the values `[5, 2, 10, 20, 16]` and the parameter `key` has the value `5`, then `numList` should contain `[2, 16]` at the end of the method and an `ArrayList` containing `[5, 10, 20]` should be returned.

Which of the following best explains why the method does not always work as intended?

(A) The method attempts to add an element to `returnList` after that element has already been removed from `numList`.

(B) The method causes a `NullPointerException` to be thrown when no matches are found.

(C) The method causes an `IndexOutOfBoundsException` to be thrown.

(D) The method fails to correctly determine whether an element of `numList` is divisible by `key`.

(E) The method skips some elements of `numList` during the traversal.

35. Consider the `mode` method, which is intended to return the most frequently occurring value (mode) in its `int[]` parameter `arr`. For example, if the parameter of the `mode` method has the contents `{6, 5, 1, 5, 2, 6, 5}`, then the method is intended to return `5`.

```
/** Precondition: arr.length >= 1 */
public static int mode(int[] arr)
{
   int modeCount = 1;
   int mode = arr[0];

   for (int j = 0; j < arr.length; j++)
   {
      int valCount = 0;
      for (int k = 0; k < arr.length; k++)
      {
         if ( /* missing condition 1 */ )
         {
            valCount++;
         }
      }
      if ( /* missing condition 2 */ )
      {
         modeCount = valCount;
         mode = arr[j];
      }
   }
   return mode;
}
```

Which of the following can replace `/* missing condition 1 */` and `/* missing condition 2 */` so the code segment works as intended?

	`/* missing condition 1 */`	`/* missing condition 2 */`
(A)	`arr[j] == arr[k]`	`valCount > modeCount`
(B)	`arr[j] == arr[k]`	`modeCount > valCount`
(C)	`arr[j] != arr[k]`	`valCount > modeCount`
(D)	`arr[j] != arr[k]`	`modeCount > valCount`
(E)	`arr[j] != arr[k]`	`modeCount != valCount`

36. Consider the following methods.

```
/** Precondition: a > 0 and b > 0 */
public static int methodOne(int a, int b)
{
   int loopCount = 0;
   for (int i = 0; i < a / b; i++)
   {
      loopCount++;
   }
   return loopCount;
}

/** Precondition: a > 0 and b > 0 */
public static int methodTwo(int a, int b)
{
   int loopCount = 0;
   int i = 0;
   while (i < a)
   {
      loopCount++;
      i += b;
   }
   return loopCount;
}
```

Which of the following best describes the conditions under which `methodOne` and `methodTwo` return the same value?

(A) When `a` and `b` are both even

(B) When `a` and `b` are both odd

(C) When `a` is even and `b` is odd

(D) When `a % b` is equal to zero

(E) When `a % b` is equal to one

37. Consider the following code segment. Assume that `num3 > num2 > 0`.

```
int num1 = 0;
int num2 = /* initial value not shown */;
int num3 = /* initial value not shown */;

while (num2 < num3)
{
   num1 += num2;
   num2++;
}
```

Which of the following best describes the contents of `num1` as a result of executing the code segment?

(A) The product of `num2` and `num3`

(B) The product of `num2` and `num3 - 1`

(C) The sum of `num2` and `num3`

(D) The sum of all integers from `num2` to `num3`, inclusive

(E) The sum of all integers from `num2` to `num3 - 1`, inclusive

38. Consider the following class definition.

```
public class Value
{
   private int num;

   public int getNum()
   {
      return num;
   }

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

The following method appears in a class other than `Value`. It is intended to sum all the `num` instance variables of the `Value` objects in its `ArrayList` parameter.

```
/** Precondition: valueList is not null */
public static int getTotal(ArrayList<Value> valueList)
{
   int total = 0;
   /* missing code */
   return total;
}
```

Which of the following code segments can replace `/* missing code */` so the `getTotal` method works as intended?

I.
```
for (int x = 0; x < valueList.size(); x++)
{
   total += valueList.get(x).getNum();
}
```

II.
```
for (Value v : valueList)
{
   total += v.getNum();
}
```

III.
```
for (Value v : valueList)
{
   total += getNum(v);
}
```

(A) I only

(B) II only

(C) III only

(D) I and II

(E) I and III

39. Consider the following recursive method.

```
public static boolean recurMethod(String str)
{
   if (str.length() <= 1)
   {
      return true;
   }
   else if (str.substring(0, 1).compareTo(str.substring(1, 2)) > 0)
   {
      return recurMethod(str.substring(1));
   }
   else
   {
      return false;
   }
}
```

Which of the following method calls will return `true` ?

(A) `recurMethod("abcba")`

(B) `recurMethod("abcde")`

(C) `recurMethod("bcdab")`

(D) `recurMethod("edcba")`

(E) `recurMethod("edcde")`

40. Consider the following class definitions.

```
public class A
{
   public String message(int i)
   {
      return "A" + i;
   }
}

public class B extends A
{
   public String message(int i)
   {
      return "B" + i;
   }
}
```

The following code segment appears in a class other than `A` or `B`.

```
A obj1 = new B();                        // Line 1
B obj2 = new B();                        // Line 2
System.out.println(obj1.message(3));     // Line 3
System.out.println(obj2.message(2));     // Line 4
```

Which of the following best explains the difference, if any, in the behavior of the code segment that will result from removing the `message` method from class `A` ?

(A) The statement in line 3 will cause a compiler error because the `message` method for `obj1` cannot be found.

(B) The statement in line 4 will cause a compiler error because the `message` method for `obj2` cannot be found.

(C) As a result of the method call in line 3, the `message` method in class `B` will be executed instead of the `message` method in class `A`.

(D) As a result of the method call in line 4, the `message` method in class `B` will be executed instead of the `message` method in class `A`.

(E) The behavior of the code segment will remain unchanged.

AP® Computer Science A Exam

SECTION II: Free Response, Questions

DO NOT OPEN THIS BOOKLET UNTIL YOU ARE TOLD TO DO SO.

At a Glance

Total Time
1 hour and 30 minutes
Number of Questions
4
Percent of Total Score
50%
Writing Instrument
Pencil
Electronic Device
None allowed
Weight
The questions are weighted equally.

Instructions

The questions for Section II are printed in this booklet. You may use the pages in this booklet to organize your answers and for scratch work, but you must write your answers in the blank space provided for each question.

The Java Quick Reference is located inside the front cover of this booklet.

Write your answer to each question in the blank space provided. Begin your response to each question at the top of a new page and completely fill in the circle at the top of each page that corresponds to the question you are answering.

All program segments must be written in Java. Show all your work. Credit for partial solutions will be given. Write clearly and legibly. Erased or crossed-out work will not be scored.

Manage your time carefully. Do not spend too much time on any one question. You may proceed freely from one question to the next. You may review your responses if you finish before the end of the exam is announced.

Java Quick Reference

Accessible methods from the Java library that may be included in the exam

Class Constructors and Methods	Explanation
String Class	
`String(String str)`	Constructs a new `String` object that represents the same sequence of characters as `str`
`int length()`	Returns the number of characters in a `String` object
`String substring(int from, int to)`	Returns the substring beginning at index `from` and ending at index `to - 1`
`String substring(int from)`	Returns `substring(from, length())`
`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found
`boolean equals(String other)`	Returns `true` if `this` is equal to `other`; returns `false` otherwise
`int compareTo(String other)`	Returns a value `<0` if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value `>0` if `this` is greater than `other`
Integer Class	
`Integer(int value)`	Constructs a new `Integer` object that represents the specified `int` value
`Integer.MIN_VALUE`	The minimum value represented by an `int` or `Integer`
`Integer.MAX_VALUE`	The maximum value represented by an `int` or `Integer`
`int intValue()`	Returns the value of this `Integer` as an `int`
Double Class	
`Double(double value)`	Constructs a new `Double` object that represents the specified `double` value
`double doubleValue()`	Returns the value of this `Double` as a `double`
Math Class	
`static int abs(int x)`	Returns the absolute value of an `int` value
`static double abs(double x)`	Returns the absolute value of a `double` value
`static double pow(double base, double exponent)`	Returns the value of the first parameter raised to the power of the second parameter
`static double sqrt(double x)`	Returns the positive square root of a `double` value
`static double random()`	Returns a `double` value greater than or equal to `0.0` and less than `1.0`
ArrayList Class	
`int size()`	Returns the number of elements in the list
`boolean add(E obj)`	Appends `obj` to end of list; returns `true`
`void add(int index, E obj)`	Inserts `obj` at position `index (0 <= index <= size)`, moving elements at position `index` and higher to the right (adds 1 to their indices) and adds 1 to size
`E get(int index)`	Returns the element at position `index` in the list
`E set(int index, E obj)`	Replaces the element at position `index` with `obj`; returns the element formerly at position `index`
`E remove(int index)`	Removes element from position `index`, moving elements at position `index + 1` and higher to the left (subtracts 1 from their indices) and subtracts 1 from size; returns the element formerly at position `index`
Object Class	
`boolean equals(Object other)`	
`String toString()`	

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 Questions booklet, but no credit will be given for anything written in this booklet. You will only earn credit for what you write in the 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. A mathematical sequence is an ordered list of numbers. This question involves a sequence called a *hailstone sequence*. If n is the value of a term in the sequence, then the following rules are used to find the next term, if one exists.

   - If n is 1, the sequence terminates.
   - If n is even, then the next term is $\frac{n}{2}$.
   - If n is odd, then the next term is $3n + 1$.

   For this question, assume that when the rules are applied, the sequence will eventually terminate with the term $n = 1$.

   The following are examples of hailstone sequences.

   Example 1: 5, 16, 8, 4, 2, 1

   - The first term is 5, so the second term is $5 * 3 + 1 = 16$.
   - The second term is 16, so the third term is $\frac{16}{2} = 8$.
   - The third term is 8, so the fourth term is $\frac{8}{2} = 4$.
   - The fourth term is 4, so the fifth term is $\frac{4}{2} = 2$.
   - The fifth term is 2, so the sixth term is $\frac{2}{2} = 1$.
   - Since the sixth term is 1, the sequence terminates.

   Example 2: 8, 4, 2, 1

   - The first term is 8, so the second term is $\frac{8}{2} = 4$.
   - The second term is 4, so the third term is $\frac{4}{2} = 2$.
   - The third term is 2, so the fourth term is $\frac{2}{2} = 1$.
   - Since the fourth term is 1, the sequence terminates.

The `Hailstone` class, shown below, is used to represent a hailstone sequence. You will write three methods in the `Hailstone` class.

```
public class Hailstone
{
    /** Returns the length of a hailstone sequence that starts with n,
     *  as described in part (a).
     *  Precondition: n > 0
     */
    public static int hailstoneLength(int n)
    {  /* to be implemented in part (a) */  }

    /** Returns true if the hailstone sequence that starts with n is considered long
     *  and false otherwise, as described in part (b).
     *  Precondition: n > 0
     */
    public static boolean isLongSeq(int n)
    {  /* to be implemented in part (b) */  }

    /** Returns the proportion of the first n hailstone sequences that are considered long,
     *  as described in part (c).
     *  Precondition: n > 0
     */
    public static double propLong(int n)
    {  /* to be implemented in part (c) */  }

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

(a) The length of a hailstone sequence is the number of terms it contains. For example, the hailstone sequence in example 1 (5, 16, 8, 4, 2, 1) has a length of 6 and the hailstone sequence in example 2 (8, 4, 2, 1) has a length of 4.

Write the method `hailstoneLength(int n)`, which returns the length of the hailstone sequence that starts with `n`.

```
/** Returns the length of a hailstone sequence that starts with n, as described in part (a).
 *  Precondition: n > 0
 */
public static int hailstoneLength(int n)
```

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 Hailstone

public static int hailstoneLength(int n)
public static boolean isLongSeq(int n)
public static double propLong(int n)
```

(b) A hailstone sequence is considered long if its length is greater than its starting value. For example, the hailstone sequence in example 1 (5, 16, 8, 4, 2, 1) is considered long because its length (6) is greater than its starting value (5). The hailstone sequence in example 2 (8, 4, 2, 1) is not considered long because its length (4) is less than or equal to its starting value (8).

Write the method `isLongSeq(int n)`, which returns `true` if the hailstone sequence starting with `n` is considered long and returns `false` otherwise. Assume that `hailstoneLength` works as intended, regardless of what you wrote in part (a). You must use `hailstoneLength` appropriately to receive full credit.

```
/** Returns true if the hailstone sequence that starts with n is considered long
 *  and false otherwise, as described in part (b).
 *  Precondition: n > 0
 */
public static boolean isLongSeq(int n)
```

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.

(c) The method `propLong(int n)` returns the proportion of long hailstone sequences with starting values between 1 and `n`, inclusive.

Consider the following table, which provides data about the hailstone sequences with starting values between 1 and 10, inclusive.

Starting Value	Terms in the Sequence	Length of the Sequence	Long?
1	1	1	No
2	2, 1	2	No
3	3, 10, 5, 16, 8, 4, 2, 1	8	Yes
4	4, 2, 1	3	No
5	5, 16, 8, 4, 2, 1	6	Yes
6	6, 3, 10, 5, 16, 8, 4, 2, 1	9	Yes
7	7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1	17	Yes
8	8, 4, 2, 1	4	No
9	9, 28, 14, 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1	20	Yes
10	10, 5, 16, 8, 4, 2, 1	7	No

The method call `Hailstone.propLong(10)` returns `0.5`, since 5 of the 10 hailstone sequences shown in the table are considered long.

Write the `propLong` method. Assume that `hailstoneLength` and `isLongSeq` work as intended, regardless of what you wrote in parts (a) and (b). You must use `isLongSeq` appropriately to receive full credit.

```
/** Returns the proportion of the first n hailstone sequences that are considered long,
 *  as described in part (c).
 *  Precondition: n > 0
 */
public static double propLong(int n)
```

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 Hailstone

public static int hailstoneLength(int n)
public static boolean isLongSeq(int n)
public static double propLong(int n)
```

NO TEST MATERIAL ON THIS PAGE

2. This question involves the creation and use of a spinner to generate random numbers in a game. A `GameSpinner` object represents a spinner with a given number of sectors, all equal in size. The `GameSpinner` class supports the following behaviors.

   - Creating a new spinner with a specified number of sectors
   - Spinning a spinner and reporting the result
   - Reporting the length of the *current run*, the number of consecutive spins that are the same as the most recent spin

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

Statements	Value Returned (blank if no value returned)	Comment
`GameSpinner g = new GameSpinner(4);`		Creates a new spinner with four sectors
`g.currentRun();`	`0`	Returns the length of the current run. The length of the current run is initially `0` because no spins have occurred.
`g.spin();`	`3`	Returns a random integer between `1` and `4`, inclusive. In this case, `3` is returned.
`g.currentRun();`	`1`	The length of the current run is `1` because there has been one spin of `3` so far.
`g.spin();`	`3`	Returns a random integer between `1` and `4`, inclusive. In this case, `3` is returned.
`g.currentRun();`	`2`	The length of the current run is `2` because there have been two `3`s in a row.
`g.spin();`	`4`	Returns a random integer between `1` and `4`, inclusive. In this case, `4` is returned.
`g.currentRun();`	`1`	The length of the current run is `1` because the spin of `4` is different from the value of the spin in the previous run of two `3`s.
`g.spin();`	`3`	Returns a random integer between `1` and `4`, inclusive. In this case, `3` is returned.
`g.currentRun();`	`1`	The length of the current run is `1` because the spin of `3` is different from the value of the spin in the previous run of one `4`.
`g.spin();`	`1`	Returns a random integer between `1` and `4`, inclusive. In this case, `1` is returned.
`g.spin();`	`1`	Returns a random integer between `1` and `4`, inclusive. In this case, `1` is returned.
`g.spin();`	`1`	Returns a random integer between `1` and `4`, inclusive. In this case, `1` is returned.
`g.currentRun();`	`3`	The length of the current run is `3` because there have been three consecutive `1`s since the previous run of one `3`.

Write the complete `GameSpinner` class. Your implementation must meet all specifications and conform to the example.

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.

3. A student plans to analyze product reviews found on a Web site by looking for keywords in posted reviews. The `ProductReview` class, shown below, is used to represent a single review. A product review consists of a product name and a review of that product.

```
public class ProductReview
{
   private String name;
   private String review;

   /** Constructs a ProductReview object and initializes the instance variables. */
   public ProductReview(String pName, String pReview)
   {
      name = pName;
      review = pReview;
   }

   /** Returns the name of the product. */
   public String getName()
   {  return name; }

   /** Returns the review of the product. */
   public String getReview()
   {  return review; }
}
```

The `ReviewCollector` class, shown below, is used to represent a collection of reviews to be analyzed.

```
public class ReviewCollector
{
   private ArrayList<ProductReview> reviewList;
   private ArrayList<String> productList;

   /** Constructs a ReviewCollector object and initializes the instance variables. */
   public ReviewCollector()
   {
      reviewList = new ArrayList<ProductReview>();
      productList = new ArrayList<String>();
   }

   /** Adds a new review to the collection of reviews, as described in part (a). */
   public void addReview(ProductReview prodReview)
   {  /* to be implemented in part (a) */  }

   /** Returns the number of good reviews for a given product name, as described in part (b). */
   public int getNumGoodReviews(String prodName)
   {  /* to be implemented in part (b) */  }

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

(a) Write the `addReview` method, which adds a single product review, represented by a `ProductReview` object, to the `ReviewCollector` object. The `addReview` method does the following when it adds a product review.

- The `ProductReview` object is added to the `reviewList` instance variable.
- The product name from the `ProductReview` object is added to the `productList` instance variable if the product name is not already found in `productList`.

Elements may be added to `reviewList` and `productList` in any order.

Complete method `addReview`.

```
/** Adds a new review to the collection of reviews, as described in part (a). */
public void addReview(ProductReview prodReview)
```

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.

(b) Write the `getNumGoodReviews` method, which returns the number of *good* reviews for a given product name. A review is considered good if it contains the string `"best"` (all lowercase). If there are no reviews with a matching product name, the method returns `0`. Note that a review that contains `"BEST"` or `"Best"` is not considered a good review (since not all the letters of `"best"` are lowercase), but a review that contains `"asbestos"` is considered a good review (since all the letters of `"best"` are lowercase).

Complete method `getNumGoodReviews`.

```
/** Returns the number of good reviews for a given product name, as described in part (b). */
public int getNumGoodReviews(String prodName)
```

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 ProductReview

private String name
private String review

public ProductReview(String pName, String pReview)
public String getName()
public String getReview()

public class ReviewCollector

private ArrayList<ProductReview> reviewList
private ArrayList<String> productList

public ReviewCollector()
public void addReview(ProductReview prodReview)
public int getNumGoodReviews(String prodName)
```

NO TEST MATERIAL ON THIS PAGE

AP Computer Science A

2020

APPENDIX - ANSWERS

Matching multiple-choice answers and free-response scoring follow.

Filed as 2019 in the source folder, but the paper itself identifies the 2020 practice exam.

Answer Key and Question Alignment to Course Framework

Multiple-Choice Question	Answer	Skill	Learning Objective	Topic
1	C	2.B	CON-1.A	Expressions and Assignment Statements
2	D	5.A	CON-2.B	Compound Boolean Expressions
			CON-2.A	if-else Statements
3	C	1.C	CON-1.D	Using the Math Class
			CON-1.A	Expressions and Assignment Statements
			CON-1.C	Casting and Ranges of Variables
4	D	2.B	CON-2.B	Compound Boolean Expressions
			CON-2.A	if Statements and Control Flow if-else Statements
5	E	1.C	MOD-1.C	Creating and Storing Objects (Instantiation)
6	A	2.A	CON-1.A	Expressions and Assignment Statements
			CON-1.C	Casting and Ranges of Variables
7	E	4.C	CON-2.B	Compound Boolean Expressions
			CON-2.A	if-else Statements
			CON-1.F	Compound Boolean Expressions
8	D	4.B	VAR-1.G	Scope and Access
			MOD-1.C	Creating and Storing Objects (Instantiation)
9	B	2.C	VAR-2.D	ArrayList Methods
10	D	4.A	CON-2.E	for Loops
			CON-1.A	Expressions and Assignment Statements
11	C	1.B	CON-2.C	while Loops
			CON-1.B	Compound Assignment Operators
12	B	1.C	CON-1.H	Comparing Objects
			CON-2.A	if Statements and Control Flow
13	E	4.C	CON-1.G	Equivalent Boolean Expressions
14	D	1.B	CON-2.A	if Statements and Control Flow else if Statements
15	E	5.A	CON-1.D	Using the Math Class
16	B	2.C	CON-2.O	Recursion
			CON-2.E	for Loops

Multiple-Choice Question	Answer	Skill	Learning Objective	Topic
17	D	2.C	MOD-3.B	Overriding Methods super Keyword
			MOD-3.D	Polymorphism
18	E	1.B	VAR-2.B	Traversing Arrays
			CON-2.E	for Loops
19	D	2.B	VAR-2.G	Traversing 2D Arrays
			VAR-2.F	2D Arrays
20	B	1.C	VAR-2.C	Enhanced for Loop for Arrays
			CON-2.I	Developing Algorithms Using Arrays
21	D	2.C	CON-2.I	Developing Algorithms Using Arrays
			VAR-1.E	String Methods
22	A	2.D	CON-2.I	Developing Algorithms Using Arrays
			VAR-1.E	String Methods
23	D	1.B	VAR-2.G	Traversing 2D Arrays
24	D	2.C	MOD-2.H	Static Variables and Methods
25	D	2.C	CON-2.F	Developing Algorithms Using Strings
			VAR-1.E	String Methods
			CON-2.E	for Loops
26	C	2.D	CON-2.H	Informal Code Analysis
			CON-2.E	for Loops
			CON-2.G	Nested Iteration
27	E	5.D	CON-1.G	Equivalent Boolean Expressions
28	B	2.C	CON-2.F	Developing Algorithms Using Strings
			VAR-1.E	String Methods
			CON-2.C	while Loops
29	E	2.C	CON-2.O	Recursion
30	B	1.C	MOD-3.B	Writing Constructors for Subclasses
31	A	1.B	VAR-2.B	Traversing Arrays
			VAR-2.C	Enhanced for Loop for Arrays
			CON-2.I	Developing Algorithms Using Arrays
32	B	2.C	MOD-3.D	Polymorphism
			MOD-2.B	Constructors
			MOD-3.B	Creating Superclasses and Subclasses
33	E	2.B	VAR-2.G	Traversing 2D Arrays

Multiple-Choice Question	Answer	Skill	Learning Objective	Topic
34	E	5.B	VAR-2.E	Traversing ArrayLists
35	A	1.B	CON-2.I	Developing Algorithms Using Arrays
36	D	5.D	CON-2.E CON-2.C CON-1.A	for Loops while Loops Expressions and Assignment Statements
37	E	5.A	CON-2.C CON-2.D	while Loops
38	D	1.C	VAR-2.D VAR-2.E	ArrayList Methods Traversing ArrayLists
39	D	4.A	CON-2.O	Recursion
40	A	5.C	MOD-3.D	Polymorphism

Free-Response Section

Scoring Guidelines

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.*

Question 1: Methods and Control Structures — 9 points

Learning Objectives: CON-1.A CON-1.C CON-1.E CON-2.A CON-2.C CON-2.E MOD-1.G MOD-2.F

Canonical solution

(a) — 3 points

```
public static int hailstoneLength(int n)
{
   int count = 1;
   while (n > 1)
   {
      if (n % 2 == 0)
      {
         n = n / 2;
      }
      else
      {
         n = 3 * n + 1;
      }
      count++;
   }
   return count;
}
```

(b) — 2 points

```
public static boolean isLongSeq(int n)
{
   return hailstoneLength(n) > n;
}
```

(c) — 4 points

```
public static double propLong(int n)
{
   int count = 0;
   for (int i = 1; i <= n; i++)
   {
      if (isLongSeq(i))
      {
         count++;
      }
   }
   return (double) count / n;
}
```

(a) `hailstoneLength`

	Scoring Criteria	Decision Rules	
1	Loops from given starting value `n` until the sequence terminates, using updated values for the current term	**Responses still earn the point even if they...** • update `n` incorrectly.	**1 point** 3.C CON-2.C
2	Computes the next value	**Responses still earn the point even if they...** • use a correct formula in an incorrect case.	**1 point** 3.C CON-1.A
3	Uses correct formula for next value depending on even/odd		**1 point** 3.C CON-2.A
		Total for part (a)	**3 points**

(b) `isLongSeq`

	Scoring Criteria	Decision Rules	
4	Calls `hailstoneLength`		**1 point** 3.A MOD-1.G
5	Correctly compares length and starting value to determine return value	**Responses still earn the point even if they...** • call `hailstoneLength` incorrectly.	**1 point** 3.C CON-1.E
		Total for part (b)	**2 points**

(c) `propLong`

	Scoring Criteria	Decision Rules	
6	Calls `isLongSeq` in the context of a loop		**1 point** 3.A MOD-1.G
7	Loops `1` to `n` (*no bounds errors*)		**1 point** 3.C CON-2.E
8	Calculates `double` proportion	**Responses still earn the point even if they...** • use incorrect values for the count of long sequences or `n`.	**1 point** 3.C CON-1.C
9	Returns correctly calculated value		**1 point** 3.B MOD-2.F
		Total for part (c)	**4 points**

Question-specific penalties
None

Total for question 1 **9 points**

Question 2: Class Design — 9 points

Learning Objectives: MOD-2.B MOD-2.D CON-1.B CON-1.D CON-2.A

Canonical solution

9 points

```
public class GameSpinner
{
   private int sectors;
   private int previousSpin = 0;
   private int currentLength = 0;

   public GameSpinner(int s)
   {
      sectors = s;
   }

   public int spin()
   {
      int newSpin = (int)(Math.random() * sectors) + 1;

      if (newSpin == previousSpin)
      {
         currentLength++;
      }
      else
      {
         previousSpin = newSpin;
         currentLength = 1;
      }
      return newSpin;

   }

   public int currentRun()
   {
      return currentLength;
   }

}
```

	GameSpinner		
	Scoring Criteria	**Decision Rules**	
1	Declares all appropriate `private` instance variables		**1 point** 3.B MOD-2.B
2	Declares method headers: `public int spin()` and `public int currentRun()`		**1 point** 3.B MOD-2.D
3	Declares header: `GameSpinner(int __)` (*must not be* `private`)		**1 point** 3.B MOD-2.B
4	Constructor initializes instance variable for number of sectors using parameter. Instance variables for previous spin and length of current run initialized correctly when declared or in constructor with default values.	**Responses still earn the point even if they...** • declare instance variables incorrectly.	**1 point** 3.B MOD-2.B
5	Computes random integer [1, number of sectors]		**1 point** 3.A CON-1.D
6	Compares new spin and last spin to determine required updates to state	**Responses still earn the point even if they...** • use an incorrectly computed random integer for new spin; or • incorrectly declare the instance variable intended to store last spin.	**1 point** 3.C CON-2.A
7	Updates instance variable that represents length of current run appropriately if new spin and previous spin are the same	**Responses still earn the point even if they...** • incorrectly compare new spin and last spin.	**1 point** 3.B MOD-2.D
8	Updates previous spin and length of current run appropriately when new spin differs from the previous spin	**Responses still earn the point even if they...** • incorrectly compare new spin and last spin.	**1 point** 3.C CON-1.B
9	`currentRun` returns updated instance variable value	**Responses still earn the point even if they...** • incorrectly update instance variables in the `spin` method.	**1 point** 3.B MOD-2.D
	Question-specific penalties		
	None		
		Total for question 2	**9 points**

Question 3: Array/ArrayList — 9 points

Learning Objectives: VAR-1.E.b VAR-2.D VAR-2.E.a MOD-1.G CON-2.F.a CON-2.J.a CON-2.K

Canonical solution

(a) — 6 points

```
public void addReview(ProductReview prodReview)
{
   reviewList.add(prodReview);

   String prodName = prodReview.getName();
   boolean found = false;
   for (String n : productList)
   {
      if (n.equals(prodName))
      {
         found = true;
      }
   }
   if (!found)
   {
      productList.add(prodName);
   }
}
```

(b) — 3 points

```
public int getNumGoodReviews(String prodName)
{
   int numGoodReviews = 0;
   for (ProductReview prodReview : reviewList)
   {
      if (prodName.equals(prodReview.getName()))
      {
         String review = prodReview.getReview();
         if (review.indexOf("best") >= 0)
         {
            numGoodReviews++;
         }
      }
   }
   return numGoodReviews;
}
```

(a) `addReview`

	Scoring Criteria	Decision Rules	
1	Adds a `ProductReview` object to `reviewList`	**Responses still earn the point even if they...** • add a `ProductReview` object other than the one referenced by the parameter `prodReview`.	**1 point** 3.D VAR-2.D
2	Gets product name of review to be added		**1 point** 3.A MOD-1.G
3	Traverses `productList` (*no bounds errors*)	**Responses still earn the point even if they...** • use a `for`, an enhanced `for`, or a `while` loop.	**1 point** 3.D VAR-2.E.a
4	Compares name in `productList` with name from review to be added	**Responses still earn the point even if they...** • use an incorrectly accessed value for either name.	**1 point** 3.C VAR-1.E.b
5	Adds new product name to `productList`	**Responses still earn the point even if they...** • add the new product name under the wrong conditions; or • add an incorrectly accessed value for the new product name	**1 point** 3.D VAR-2.D
6	Correctly adds product name to `productList` if and only if the product name is not already in `productList`		**1 point** 3.D CON-2.K
		Total for part (a)	**6 points**

(b) `getNumGoodReviews`

	Scoring Criteria	Decision Rules	
7	Traverses `reviewList` (*no bounds errors*)	**Responses still earn the point even if they...** • use a `for`, an enhanced `for`, or a `while` loop.	**1 point** 3.D VAR-2.E.a
8	Selects all and only reviews with matching product names that contain `"best"`		**1 point** 3.C CON-2.F.a
9	Returns correct count of good reviews		**1 point** 3.D CON-2.J.a
		Total for part (b)	**3 points**
	Question-specific penalties		
	None		
		Total for question 3	**9 points**

Question 4: 2D Array — 9 points

Learning Objectives: MOD-1.D.b MOD-1.G CON-1.H CON-2.A CON-2.N.c VAR-2.F VAR-2.G.a

Canonical solution

(a) — 5 points

```
public Theater(int seatsPerRow, int tier1Rows,
               int tier2Rows)
{
   theaterSeats =
      new Seat[tier1Rows + tier2Rows][seatsPerRow];
   for (int r = 0; r < tier1Rows + tier2Rows; r++)
   {
      for (int c = 0; c < seatsPerRow; c++)
      {
         if (r < tier1Rows)
         {
            theaterSeats[r][c] = new Seat(true, 1);
         }
         else
         {
            theaterSeats[r][c] = new Seat(true, 2);
         }
      }
   }
}
```

(b) — 4 points

```
public boolean reassignSeat(int fromRow, int fromCol,
                            int toRow, int toCol)
{
   Seat toS = theaterSeats[toRow][toCol];
   if (!toS.isAvailable())
   {
      return false;
   }

   Seat fromS = theaterSeats[fromRow][fromCol];
   if (toS.getTier() < fromS.getTier())
   {
      return false;
   }

   toS.setAvailability(false);
   fromS.setAvailability(true);
   return true;
}
```

(a) `Theater`

	Scoring Criteria	Decision Rules	
1	Instantiates a new `Seat[][]` with the correct number of rows and columns, based on parameters		**1 point** 3.E VAR-2.F
2	Traverses the `theaterSeats` array (*no bounds errors*)		**1 point** 3.E VAR-2.G.a
3	Instantiates a new `Seat` object with a tier and availability status	**Responses still earn the point even if they...** • incorrectly assign the new object to a `theaterSeats` element.	**1 point** 3.A MOD-1.D.b
4	Accesses a `theaterSeats` element and assigns it a new `Seat` object	**Responses still earn the point even if they...** • incorrectly instantiate the new `Seat` object; or • assign the new `Seat` object to an incorrect `theaterSeats` element.	**1 point** 3.E VAR-2.F
5	Correct tiers assigned to all array elements		**1 point** 3.C CON-2.A
		Total for part (a)	**5 points**

(b) `reassignSeat`

	Scoring Criteria	Decision Rules	
6	Accesses *from* and *to* `Seat` objects		**1 point** 3.E VAR-2.F
7	Calls `isAvailable` and `getTier` on `Seat` objects	**Responses still earn the point even if they...** • correctly call methods on `theaterSeats` elements other than the *to* and *from* seats.	**1 point** 3.A MOD-1.G
8	Checks if move can be made based on both tiers and the availability status of *to* `Seat` object		**1 point** 3.C CON-1.H
9	Correctly updates availability of both seats and returns `true` if the move can be made; otherwise, returns `false`		**1 point** 3.E CON-2.N.c
		Total for part (b)	**4 points**
	Question-specific penalties		
	None		
		Total for question 4	**9 points**