Quiz Review 1

1. Consider the following method.

```
/** Precondition: arr contains only positive values.
 */
public static void doSome(int[] arr, int lim)
{
  int v = 0;
  int k = 0;
  while (k < arr.length && arr[k] < lim)
  {
    if (arr[k] > v)
    {
      v = arr[k]; /* Statement S */
    }
    k++; /* Statement T */
  }
}
```

Assume that doSome is called and executes without error. Which of the following are possible combinations for the value of lim, the number of times *Statement* S is executed, and the number of times *Statement* T is executed?

	Value of `lim`	Executions of *Statement S*	Executions of *Statement T*
I.	5	0	5
II.	7	4	9
III.	3	5	2

(A) I only

(B) II only ✓

(C) III only

(D) I and III only

(E) II and III only

Quiz Review 1

2. In the code segment below, assume that the `int` variable `n` has been properly declared and initialized. The code segment is intended to print a value that is `1` more than twice the value of `n`.

```
/* missing code */
System.out.print(result);
```

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

I.
```
int result = 2 * n;
result = result + 1;
```

II.
```
int result = n + 1;
result = result * 2;
```

III.
```
int result = (n + 1) * 2;
```

(A) I only ✓

(B) II only

(C) III only

(D) I and III

(E) II and III

Answer A

Correct. The correct answer is I only. Statement I is correct because the value of `result` at the end of the code segment is `2 * n + 1`, which is the intended value. Statements II and III are incorrect because the value of `result` at the end of each code segment is `(n + 1) * 2`, which is different than the intended value `2 * n + 1`.

3. The `Date` class below will contain three `int` attributes for day, month, and year, a constructor, and a `setDate` method. The `setDate` method is intended to be accessed outside the class.

```
public class Date
{
     /* missing code */
}
```

Which of the following replacements for `/* missing code */` is the most appropriate implementation of the class?

Quiz Review 1

(A)
```
private int day;
private int month;
private int year;
private Date()
{ /* implementation not shown */ }
private void setDate(int d, int m, int y)
{ /* implementation not shown */ }
```

(B)
```
private int day;
private int month;
private int year;
public Date()
{ /* implementation not shown */ }
private void setDate(int d, int m, int y)
{ /* implementation not shown */ }
```

(C) ✓
```
private int day;
private int month;
private int year;
public Date()
{ /* implementation not shown */ }
public void setDate(int d, int m, int y)
{ /* implementation not shown */ }
```

(D)
```
public int day;
public int month;
public int year;
private Date()
{ /* implementation not shown */ }
private void setDate(int d, int m, int y)
{ /* implementation not shown */ }
```

(E)
```
public int day;
public int month;
public int year;
public Date()
{ /* implementation not shown */ }
public void setDate(int d, int m, int y)
{ /* implementation not shown */ }
```

Answer C

Correct. The instance variables `day`, `month`, and `year` should be designated as `private` so they are kept internal to the class. The constructor should be designated as `public` so that `Date` objects can be created outside the class. The `setDate` method should be designated as `public`, as it is intended to be accessed from outside the class.

Quiz Review 1

4. The `Player` class below will contain two `int` attributes and a constructor. The class will also contain a method `getScore` that can be accessed from outside the class.

```
public class Player
{
     /* missing code */
}
```

Which of the following replacements for `/* missing code */` is the most appropriate implementation of the class?

(A)
```
private int score;
private int id;
private Player(int playerScore, int playerID)
{ /* implementation not shown */ }
private int getScore()
{ /* implementation not shown */ }
```

(B)
```
private int score;
private int id;
public Player(int playerScore, int playerID)
{ /* implementation not shown */ }
private int getScore()
{ /* implementation not shown */ }
```

(C) ✓
```
private int score;
private int id;
public Player(int playerScore, int playerID)
{ /* implementation not shown */ }
public int getScore()
{ /* implementation not shown */ }
```

(D)
```
public int score;
public int id;
public Player(int playerScore, int playerID)
{ /* implementation not shown */ }
private int getScore()
{ /* implementation not shown */ }
```

(E)
```
public int score;
public int id;
public Player(int playerScore, int playerID)
{ /* implementation not shown */ }
public int getScore()
{ /* implementation not shown */ }
```

Answer C

Correct. The instance variables `score` and `id` should be designated as `private` so they are kept internal to the class. The constructor should be designated as public so that `Player` objects can be created outside the class. The `getScore` method should be designated as `public`, as it is intended

Quiz Review 1

to be accessible from outside the class.

5. Consider the following code segment, which is intended to find the average of two positive integers, `x` and `y`.

```
int x;
int y;
int sum = x + y;
double average = (double) (sum / 2);
```

Which of the following best describes the error, if any, in the code segment?

(A) There is no error, and the code works as intended.

(B) In the expression `(double) (sum / 2)`, the cast to `double` is applied too late, so the average will be less than the expected result for even values of `sum`.

(C) In the expression `(double) (sum / 2)`, the cast to `double` is applied too late, so the average will be greater than the expected result for even values of `sum`.

(D) In the expression `(double) (sum / 2)`, the cast to `double` is applied too late, so the average will be less than the expected result for odd values of `sum`. ✓

(E) In the expression `(double) (sum / 2)`, the cast to `double` is applied too late, so the average will be greater than the expected result for odd values of `sum`.

Answer D

Correct. Since `sum` is declared as an `int` type and `2`, by default, is also an `int` type, the division will result in an `int` type causing a loss of precision whenever an odd number is divided by an even number. Casting the expression `sum / 2` to `double` after the division has occurred is too late. It is not possible to recover that loss of precision. So for odd values of `sum`, the value of `average` will be less than the expected average.

6. 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` ✓

Quiz Review 1

Answer E

Correct. By De Morgan's laws, `!(a && b)` is equivalent to `!a || !b` and the entire expression is equivalent to `!a || !b || c`.

7. Assume that the boolean variables a, b, c, and d have been declared and initialized. Consider the following expression.

!(!(a && b) || (c || !d))

Which of the following is equivalent to the expression?

(A) (a && b) && (!c && d) ✓

(B) (a || b) && (!c && d)

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

(D) (!a || !b) && (!c && d)

(E) !(a && b) && (c || !d)

8. Consider the following static method.

```
public static int calculate(int x)
{
  x = x + x;
  x = x + x;
  x = x + x;

  return x;
}
```

Which of the following can be used to replace the body of calculate so that the modified version of calculate will return the same result as the original version for all x ?

Quiz Review 1

(A) return 3 + x;
(B) return 3 * x;
(C) return 4 * x;
(D) return 6 * x;
(E) return 8 * x; ✓

9. Consider the following code segment.

```
double num = 9 / 4;
System.out.print(num);
System.out.print(" ");
System.out.print((int) num);
```

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

(A) `2 2`
(B) `2.0 2` ✓
(C) `2.0 2.0`
(D) `2.25 2`
(E) `2.25 2.0`

Answer B

Correct. The value of `num` is `2.0`, not `2.25` since `9` and `4` are by default `int` types. This results in integer division rather than floating point division. So the first `System.out.print` statement will display `2.0`. The second `System.out.print` statement will display a space. In the third `System.out.print` statement, `num` is cast to an `int` type, so the statement will display `2`.

10. 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

Quiz Review 1

Answer A

Correct. In option I, the cast applies to the value `2`, so floating-point division is performed and the expression evaluates to `0.5 + 3`, or `3.5`. In option II, the cast applies to the result of the integer division `2 / 4`, so the expression evaluates to `0.0 + 3`, or `3.0`. In option III, the cast applies to the sum of `3` and the result of the integer division `2 / 4`, so the expression evaluates to `(double) (0 + 3)`, or `3.0`.

11. Consider the following code segment.

```
int w = 1;
int x = w / 2;
double y = 3;
int z = (int) (x + y);
```

Which of the following best describes the results of compiling the code segment?

(A) The code segment compiles without error. ✓

(B) The code segment does not compile, because the `int` variable `x` cannot be assigned the result of the operation `w / 2`.

(C) The code segment does not compile, because the integer value `3` cannot be assigned to the `double` variable `y`.

(D) The code segment does not compile, because the operands of the addition operator cannot be of different types `int` and `double`.

(E) The code segment does not compile because the result of the addition operation is of type `double` and cannot be cast to an `int`.

Answer A

Correct. The first statement assigns an integer value to an `int` variable. The second statement performs integer division and assigns the `int` result `0` to an `int` variable. In the third statement, the integer value `3` is cast to a `double` and assigned to a `double` variable. In the final statement, the sum of `x` and `y` is a `double`, since at least one of the operands is a `double`. The `double` result is cast to an `int` and stored in an `int` variable.

Quiz Review 1

12. Consider the following method.

```
/** Precondition: bound >= 0 */
public int sum(int bound)
{
     int answer = 0;
     for (int i = 0; i < bound; i++)
     {
         answer += bound;
     }
     return answer;
}
```

Assume that `sum` is called with a parameter that satisfies the precondition and that it executes without error. How many times is the test expression `i < bound` in the `for` loop header evaluated?

(A) `0`

(B) `bound - 1`

(C) `bound`

(D) `bound + 1` ✓

(E) An unknown number of times

Answer D

Correct. When `bound` is `0`, the Boolean expression in the `for` loop header is evaluated once and is found to be `false`, so the loop body is never executed. When `bound` is `1`, the Boolean expression is evaluated twice—the first time it evaluates to `true` and the second time it evaluates to `false`. In general, the Boolean expression is evaluated `bound + 1` times—the first `bound` times, when it evaluates to `true` and the loop body is executed, and the last time, when it evaluates to `false` and the loop terminates.

Quiz Review 1

13. The following statement assigns an integer value to `x`.

```
int x = (int)(Math.random() * 5) + 10;
```

Consider the statement that would result if the positions of `5` and `10` were swapped in the preceding statement and the resulting integer were assigned to `y`.

```
int y = (int)(Math.random() * 10) + 5;
```

Which of the following are true statements about how the possible values assigned to `y` differ from the possible values assigned to `x` ?

I. There are more possible values of `x` than there are possible values of `y`.
II. There are more possible values of `y` than there are possible values of `x`.
III. The value assigned to `x` can sometimes be the same as the value assigned to `y`.

(A) I only
(B) II only
(C) III only
(D) I and III
(E) II and III ✓

Answer E

Correct. The variable `x` takes on one of 5 possible values, between `10` and `14`, inclusive. The variable `y` takes on one of 10 possible values, between `5` and `14`, inclusive. Statement I is false. Statement II is true. Statement III is true because both `x` and `y` can sometimes be assigned a value between `10` and `14`, inclusive.

14. 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.

Quiz Review 1

15. Assume that x and y have been declared and initialized with int values. Consider the following Java expression.

```
(y > 10000) || (x > 1000 && x < 1500)
```

Which of the following is equivalent to the expression given above?

(A) (y > 10000 | | x > 1000) && (y > 10000 | | x < 1500) ✓

(B) (y > 10000 | | x > 1000) | | (y > 10000 | | x < 1500)

(C) (y > 10000) && (x > 1000 | | x < 1500)

(D) (y > 10000 && x > 1000) | | (y > 10000 && x < 1500)

(E) (y > 10000 && x > 1000) && (y > 10000 && x < 1500)

16. Assume that a, b, and c are variables of type int. Consider the following three conditions.

I. (a == b) && (a == c) && (b == c)

II. (a == b) || (a == c) || (b == c)

III. ((a - b) * (a - c) * (b - c)) == 0

Assume that subtraction and multiplication never overflow. Which of the conditions above is (are) always true if at least two of a, b, and c are equal?

(A) I only

(B) II only

(C) III only

(D) I and II

(E) II and III ✓

17. Assume that x and y are boolean variables and have been properly initialized.

(x && y) || !(x && y)

The result of evaluating the expression above is best described as

(A) always true ✓

(B) always false

(C) true only when x is true and y is true

(D) true only when x and y have the same value

(E) true only when x and y have different values

Quiz Review 1

18. Consider the following method.

```
public void conditionalTest(int a, int b)
{
  if ((a > 0) && (b > 0))
  {
    if (a > b)
      System.out.println("A");
    else
      System.out.println("B");
  }
  else if ((b < 0) || (a < 0))
    System.out.println("C");
  else
    System.out.println("D");
}
```

What is printed as a result of the call conditionalTest(3, -2)?

(A) A

(B) B

(C) C ✓

(D) D

(E) Nothing is printed.

Quiz Review 1

19. 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

Answer A

Correct. In code segment I, `i` takes on the values `-1` through `nums.length - 2`, inclusive, in the `while` loop. Since `i` is incremented before the `if` statement, the array elements `nums[0]` through `nums[nums.length - 1]` are compared to `0`. In code segment II, array element

Quiz Review 1

`nums[0]` is excluded since the first iteration of the `for` loop accesses `nums[1]`. In code segment III, the variable `i` represents an element of the array rather than an index.

20. Consider the definition of the `Person` class below. The class uses the instance variable `adult` to indicate whether a person is an adult or not.

```
public class Person

{
     private String name;
     private int age;
     private boolean adult;
     public Person (String n, int a)
     {
          name = n;
          age = a;
          if (age >= 18)
          {
             adult = true;
          }
          else
          {
             adult = false;
          }
     }
}
```

Which of the following statements will create a `Person` object that represents an adult person?

(A) `Person p = new Person ("Homer", "adult");`

(B) `Person p = new Person ("Homer", 23);` ✓

(C) `Person p = new Person ("Homer", "23");`

(D) `Person p = new Person ("Homer", true);`

(E) `Person p = new Person ("Homer", 17);`

Answer B

Correct. The `Person` class constructor requires a `String` parameter and an `int` parameter; the person will correctly be designated as an adult because the `age` parameter is greater than `18`.

Quiz Review 1

21. Consider the following two code segments. Code segment II is a revision of code segment I in which the loop header has been changed.

I.

```
for (int k = 1; k <= 5; k++)
{
     System.out.print(k);
}
```

II.

```
for (int k = 5; k >= 1; k--)
{
     System.out.print(k);
}
```

Which of the following best explains how the output changes from code segment I to code segment II?

(A) Both code segments produce the same output, because they both iterate four times.

(B) Both code segments produce the same output, because they both iterate five times.

(C) Code segment I prints more values than code segment II does, because it iterates for one additional value of `k`.

(D) Code segment II prints more values than code segment I, because it iterates for one additional value of `k`.

(E) The code segments print the same values but in a different order, because code segment I iterates from `1` to `5` and code segment II iterates from `5` to `1`. ✓

Answer E

Correct. Code segment I iterates from `1` to `5`, producing the output `12345`. Code segment II iterates from `5` to `1`, producing the output `54321`.

22. Consider the following code segment.

```
if (a < b || c != d)
{
     System.out.println("dog");
}
else
{
     System.out.println("cat");
}
```

Assume that the `int` variables `a`, `b`, `c`, and `d` have been properly declared and initialized. Which of the following code segments produces the same output as the given code segment for all values of `a`, `b`, `c`, and `d` ?

Quiz Review 1

(A)
```
if (a < b && c != d)
{
     System.out.println("dog");
}
else
{
     System.out.println("cat");
}
```

(B)
```
if (a < b && c != d)
{
     System.out.println("cat");
}
else
{
     System.out.println("dog");
}
```

(C)
```
if (a > b && c == d)
{
     System.out.println("cat");
}
else
{
     System.out.println("dog");
}
```

(D)
```
if (a >= b || c == d)
{
     System.out.println("cat");
}
else
{
     System.out.println("dog");
}
```

(E) ✓
```
if (a >= b && c == d)
{
     System.out.println("cat");
}
else
{
     System.out.println("dog");
}
```

Answer E

Correct. Since the original expression prints `"dog"` when `a < b || c != d` evaluates to `true`, an equivalent code segment would print `"cat"` when the negative of the expression, or `!(a < b || c != d)`, evaluates to `true`. Applying De Morgan's laws produces the equivalent Boolean expression `a >= b && c == d` for the case when `"cat"` should be printed.

Quiz Review 1

23. Assume that `a`, `b`, `c`, and `d` have been declared and initialized with `int` values.

```
!((a >= b) && !(c < d))
```

Which of the following is equivalent to the expression above?

(A) `(a < b) || (c < d)` ✓

(B) `(a < b) || (c >= d)`

(C) `(a < b) && (c < d)`

(D) `(a >= b) || (c < d)`

(E) `(a >= b) && (c < d)`

Answer A

Correct. By De Morgan's laws, the given expression is equivalent to `(a < b) || (c < d)`.

24. 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`

25. 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` ?

Quiz Review 1

(A) `private int getX()`

(B) `private void getX()`

(C) `public int getX()` ✓

(D) `public void getX()`

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

26. Consider the following code segment.

```
int count = 0;
for (int k = 0; k < 10; k++)
{
     count++;
}
System.out.println(count);
```

Which of the following code segments will produce the same output as the code segment above?

(A)
```
int count = 0;
for (int k = 1; k < 10; k++)
{
     count++;
}
System.out.println(count);
```

(B)
```
int count = 1;
for (int k = 1; k <= 10; k++)
{
     count++;
}
System.out.println(count);
```

(C)
```
int count = 1;
for (int k = 0; k <= 9; k++)
{
     count++;
}
System.out.println(count);
```

(D) ✓
```
int count = 0;
for (int k = 9; k >= 0; k--)
{
     count++;
}
System.out.println(count);
```

(E)
```
int count = 0;
for (int k = 10; k >= 0; k--)
{
     count++;
}
System.out.println(count);
```

Quiz Review 1

Answer D

Correct. The loop iterates 10 times. In each iteration, `count` is incremented by `1`. Since the initial value of `count` is `0`, the printed value of `count` is `10`.

27. Consider the following class definition. The class does not compile.

```
public class Player
{
     private double score;
     public getScore()
     {
         return score;
     }
     // Constructor not shown
}
```

The accessor method `getScore` is intended to return the score of a `Player` object. Which of the following best explains why the class does not compile?

(A) The `getScore` method should be declared as `private`.

(B) The `getScore` method requires a parameter.

(C) The return type of the `getScore` method needs to be defined as `double`. ✓

(D) The return type of the `getScore` method needs to be defined as `String`.

(E) The return type of the `getScore` method needs to be defined as `void`.

Answer C

Correct. The accessor method `getScore` is intended to return the value of the `double` instance variable `score`, so it should be defined as type `double`.

Quiz Review 1

28. 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);` ✓

Answer E

Correct. In the body of the first `if` clause in the code segment, `b1` retains the value `true` if `num` is between `0` and `100`, exclusive. In the body of the `else` clause, `b1` retains the value `true` if `num` is less than `-100`. The statement assigns `true` to `b2` if `num` is less than `-100` or between `0` and `100`, exclusive.

Quiz Review 1

29. Consider the following code segment.

```
double firstDouble = 2.5;
int firstInt = 30;
int secondInt = 5;
double secondDouble = firstInt - secondInt / firstDouble + 2.5;
```

What value will be assigned to `secondDouble` when the code segment is executed?

(A) `5.0`

(B) `12.5`

(C) `25.5`

(D) `29.0`

(E) `30.5` ✓

Answer E

Correct. Division has higher precedence than addition and subtraction, which have the same precedence. Operators with the same precedence are evaluated in left-to-right order. The statement that assigns a value to `secondDouble` evaluates `secondInt / firstDouble` first, resulting in `2.0`. The expression `firstInt - 2.0` is evaluated next, resulting in `28.0`. Finally, the expression `28.0 + 2.5` is evaluated resulting in `30.5`.

30. Consider the following code segment, which is intended to print the digits of the two-digit `int` number `num` in reverse order. For example, if `num` has the value `75`, the code segment should print `57`. Assume that `num` has been properly declared and initialized.

```
/* missing code */
System.out.print(onesDigit);
System.out.print(tensDigit);
```

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

Quiz Review 1

(A)
```
int onesDigit = num % 10;
int tensDigit = num / 10;
```
✓

(B)
```
int onesDigit = num / 10;
int tensDigit = num % 10;
```

(C)
```
int onesDigit = 10 / num;
int tensDigit = 10 % num;
```

(D)
```
int onesDigit = num % 100;
int tensDigit = num / 100;
```

(E)
```
int onesDigit = num / 100;
int tensDigit = num % 100;
```

Answer A

Correct. The expression `num % 10` extracts the rightmost digit of `num` by evaluating to the remainder when `num` is divided by `10`. The expression `num / 10` extracts the leftmost digit by evaluating to the result of the integer division of `num` by `10`.

Quiz Review 1

31. Consider the following method, isSorted, which is intended to return true if an array of integers is sorted in nondecreasing order and to return false otherwise.

```
/** @param data an array of integers
 *  @return true if the values in the array appear in sorted (nondecreasing) order
 */
public static boolean isSorted(int[] data)
{
  /* missing code */
}
```

Which of the following can be used to replace /* missing code */ so that isSorted will work as intended?

I.
```
for (int k = 1; k < data.length; k++)
{
  if (data[k - 1] > data[k])
    return false;
}
return true;
```

II.
```
for (int k = 0; k < data.length; k++)
{
  if (data[k] > data[k + 1])
    return false;
}
return true;
```

III.
```
for (int k = 0; k < data.length - 1; k++)
{
  if (data[k] > data[k + 1])
    return false;
  else
    return true;
}
return true;
```

(A) I only ✓

(B) II only

(C) III only

(D) I and II only

(E) I and III only

Quiz Review 1

Answer A

Correct. Choice I has a loop control variable k that starts at 1, increments by 1, and will terminate the loop when k has the value data.length. In each iteration, there is a check to see if the value before the current value is larger. If it is, false is returned because the elements would not be nondecreasing. For example, if data[k – 1] had the value 5 and data[k] had the value 4, then data would contain an instance where the values were decreasing. If all the elements are checked and no decreasing pairs of elements are found (the for loop ends without returning), the method will return true. Choice II has a loop control variable k that starts at 0, increments by 1, and will terminate the loop when k has the value data.length. In each iteration, there is a check to see if the current value is larger than the subsequent value. If it is, false is returned because elements would not be nondecreasing. For example, if data[k] had the value 5 and data[k + 1] had the value 4, then data would contain an instance where the values were decreasing. Unfortunately, since the indices of an array start at 0 and go through data.length – 1, when k has the value data.length – 1 an ArrayIndexOutOfBoundsException will be thrown as the condition attempts to check data[data.length – 1] and data[data.length – 1 + 1] or data[data.length]. Choice III has a loop control variable k that starts at 0, increments by 1, and will terminate the loop when k has the value data.length – 1. In each iteration, there is a check to see if the current value is larger than the subsequent value. If it is, false is returned because elements would not be nondecreasing, otherwise true is returned. As a result, only data[0] and data[1] are examined. The remaining elements in data are not checked because the method will stop once a return statement is reached. This means that the method could return true even when there are consecutive elements in data that are nondecreasing.

32. The following code segment is intended to interchange the values of the `int` variables `x` and `y`. Assume that `x` and `y` have been properly declared and initialized.

```
int temp = x;
/* missing code */
```

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

(A)
```
x = y;
x = temp;
```

(B) ✓
```
x = y;
y = temp;
```

(C)
```
y = x;
x = temp;
```

(D)
```
y = x;
temp = y;
```

(E)
```
y = x;
temp = x;
```

Answer B

Correct. Since the original value of `x` has been stored in `temp`, the variable `x` can be assigned the value of `y` and then `y` can be assigned the original value of `x`, as stored in `temp`.

Quiz Review 1

33. 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`

Answer C

Correct. In the first assignment statement, the multiplication operation is evaluated before the addition operation and `a` is assigned the value `9`. In the second assignment statement, the integer division is evaluated first and produces a result of `1`, which is added to `4` so that the variable `b` is assigned the value `5`. In the third assignment statement, the remainder operation is evaluated before the addition operation and `c` is assigned the value `6`. The variable `d` is assigned the value `20.0`.

34. Assume that the following variable declarations have been made.

```
double d = Math.random();

double r;
```

Which of the following assigns a value to `r` from the uniform distribution over the range $0.5 \leq r < 5.5$?

(A) `r = d + 0.5;`

(B) `r = d + 0.5 * 5.0;`

(C) `r = d * 5.0;`

(D) `r = d * 5.0 + 0.5;` ✓

(E) `r = d * 5.5;`

Quiz Review 1

35. Which of the following code segments will print all multiples of 5 that are greater than 0 and less than 100 ?

```
I. for (int k = 1; k < 100; k++)
{
   if (k % 5 == 0)
   {
      System.out.print(k + " ");
   }
}

II. for (int k = 1; k < 100; k++)
{
   if (k / 5 == 0)
   {
      System.out.print(k + " ");
   }
}

III. int k = 5;
while (k < 100)
{
   System.out.print(k + " ");
   k = k + 5;
}
```

(A) I only
(B) II only
(C) III only
(D) I and III ✓
(E) II and III

Answer D

Correct. Code segment I is correct. The variable `k` takes on values from `1` to `99`, inclusive, in the `for` loop. The value of `k` is printed when `k % 5` is zero, which happens when `k` is `5`, `10`, `15`, `20`, ..., and `95`. Code segment II is incorrect. The variable `k` takes on values from `1` to `99`, inclusive, in the `for` loop. The value of `k` is printed when `k / 5` is zero, which only happens when `k` is `1`, `2`, `3`, and `4`. When `k` is `20`, for example, `k / 5 == 0` is equivalent to `4 == 0`, which is `false`, so `20` is not printed. Code segment III is correct. The variable `k` takes on values `5`, `10`, `15`, `20`, ..., `95`, inclusive, in the `while` loop, and each value of `k` is printed.

Quiz Review 1

36. Consider the following code segment.

```
for (int k = 1; k <= 100; k++)
  if ((k % 4) == 0)
    System.out.println(k);
```

Which of the following code segments will produce the same output as the code segment above?

(A)
```
for (int k = 1; k <= 25; k++)
  System.out.println(k);
```

(B)
```
for (int k = 1; k <= 100; k = k + 4)
  System.out.println(k);
```

(C)
```
for (int k = 1; k <= 100; k++)
  System.out.println(k % 4);
```

(D)
```
for (int k = 4; k <= 25; k = 4 * k)
  System.out.println(k);
```

(E)
```
for (int k = 4; k <= 100; k = k + 4)
  System.out.println(k);
```
✓

37. Consider the following code segment.

```
for (int j = 1; j < 10; j += 2)
{
     System.out.print(j);
}
```

Which of the following code segments will produce the same output as the code segment above?

Quiz Review 1

(A)
```
int j = 1;
while (j < 10)
{
      j += 2;
      System.out.print(j);
}
```

(B) ✓
```
int j = 1;
while (j < 10)
{
      System.out.print(j);
      j += 2;
}
```

(C)
```
int j = 1;
while (j <= 10)
{
      j += 2;
      System.out.print(j);
}
```

(D)
```
int j = 1;
while (j >= 10)
{
      j += 2;
      System.out.print(j);
}
```

(E)
```
int j = 1;
while (j >= 10)
{
      System.out.print(j);
      j += 2;
}
```

Answer B

Correct. In the given `for` loop, `j` is initially `1` and increases by `2` repeatedly as long as it is less than `10`. In this `while` loop, `j` is initially `1` and increases by `2` repeatedly while it is less than `10`. Both code segments produce the output `13579`.

38. Which of the following statements stores the value `3` in `x` ?

(A) `int x = 4 / 7;`

(B) `int x = 7 / 3;`

(C) `int x = 7 / 4;`

(D) `int x = 5 % 8;`

(E) `int x = 8 % 5;` ✓

Quiz Review 1

Answer E

Correct. The expression `8 % 5` evaluates to the remainder left when `8` is divided by `5`, or `3`.

Quiz Review 1

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

Notes:

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

4. This question involves analyzing a salesperson's sales. Sales amounts are obtained using the `getSales` method in the following `SalesSimulator` class. You will write one method in the class.

```
public class SalesSimulator
{
 /** Simulates and returns the sales, in dollars, made by a salesperson
 on a particular
  *  day
  */
 public static int getSales()
 {  /* implementation not shown */  }

 /** Analyzes sales for numDays days obtained from the getSales method
 and
   *   returns the total bonus earned by the salesperson, in dollars, as
 described in part (a)
  *  Precondition: goal > 0, numDays > 0
  */
 public static int calculateBonus(int goal, int numDays)
 {  /* to be implemented in part (a) */  }

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

(a) Write the `SalesSimulator` method `calculateBonus`, which obtains the daily sales of a salesperson and applies a bonus based on how close to meeting or exceeding the daily sales goal the salesperson came. It obtains daily sales for `numDays` days using the `getSales` method. For each day, if the daily sales are at least 80 percent of `goal` but less than `goal`, the salesperson receives a $50 bonus. The salesperson receives a $75 bonus if the daily sales are greater than or equal to `goal`. Method `calculateBonus` returns the total bonus received, in dollars, over all days.

The following table shows the bonuses received as a result of the method call

Quiz Review 1

`SalesSimulator.calculateBonus(200, 3)`.

Day	`getSales()` Return Value	Daily Bonus	Explanation
1	`180`	`50`	Because $180 is greater than or equal to 80 percent of the goal ($160) but is less than the goal, the salesperson earns a $50 bonus.
2	`150`	`0`	Because $150 is less than 80 percent of the goal ($160), the salesperson earns no bonus.
3	`200`	`75`	Because $200 is greater than or equal to the goal, the salesperson earns a $75 bonus.

For the sales shown in the table above, `SalesSimulator.calculateBonus(200, 3)` should return the total bonus `125`.

Complete method `calculateBonus`.

```
/** Analyzes sales for numDays days obtained from the getSales method and returns
  *   the total bonus earned by the salesperson, in dollars, as described in part (a)
 *  Precondition: goal > 0, numDays > 0
 */
public static int calculateBonus(int goal, int numDays)
```

(b) A programmer wants to modify the `SalesSimulator` class so that the daily goal is maintained in a variable with a value that cannot be changed once it is initialized.

Write a description of how you would change the `SalesSimulator` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

Part A – calculateBonus

Select a point value to view scoring criteria, solutions, and/or examples and to score the response. +1 indicates a point earned and -1 indicates a general or question-specific penalty.

Quiz Review 1

+1 [Skill 3.C] Iterates numDays times

Responses can still earn the point even if they return early inside the loop

+1 [Skill 3.A] Calls getSales

Responses will not earn the point if they call getSales incorrectly

+1 [Skill 3.C] Compares getSales return value with goal to determine bonus categories

Responses will not earn the point if they:

· do not change bonus based on the selected category

· classify a sales amount into the wrong category, or into both categories

· call getSales more than one time to make condition comparisons

+1 [Skill 3.C] Declares, initializes, and increments bonus appropriately in all cases (algorithm)

Responses can still earn the point even if they classify a sales amount into the wrong category, or into both categories

Responses will not earn the point if they:

· do not include a loop

· return early inside the loop

+1 [Skill 3.A] Returns identified bonus

Responses can still earn the point even if they determine bonus incorrectly

-1 (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)

-1 (x) Local variables used but none declared

Canonical Solution:

Quiz Review 1

```
public static int calculateBonus(int goal,
                                 int numDays)
{
   int bonus = 0;
   for (int i = 0; i < numDays; i++)
   {
      double sales = getSales();
      if (sales >= goal)
      {
         bonus += 75;
      }
      else if (sales >= 0.8 * goal)
      {
         bonus += 50;
      }

   }
   return bonus;
}
```

= goal) Line 8: { Line 9: bonus += 75; Line 10: } Line 11: else if (sales >= 0.8 * goal) Line 12: { Line 13: bonus += 50; Line 14: } Line 15 is blank. Line 16: } Line 17: return bonus; Line 18: } end code">

0	1	2	3	4	5 ✓

Total number of points earned (minus penalties) is equal to 5.

- ☐ **+1** Iterates numDays times **(Points earned)**
- ☐ **+1** Calls getSales **(Points earned)**
- ☐ **+1** Compares getSales return value with goal to determine bonus categories **(Points earned)**
- ☐ **+1** Declares, initializes, and increments bonus appropriately in all cases (algorithm) **(Points earned)**
- ☐ **+1** Returns identified bonus **(Points earned)**
- ☐ **-1** [penalty] (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check) **(General penalties)**
- ☐ **-1** [penalty] (x) Local variables used but none declared **(General penalties)**

Canonical Solution:

Quiz Review 1

```
public static int calculateBonus(int goal,
                                 int numDays)
{
   int bonus = 0;
   for (int i = 0; i < numDays; i++)
   {
      double sales = getSales();
      if (sales >= goal)
      {
         bonus += 75;
      }
      else if (sales >= 0.8 * goal)
      {
         bonus += 50;
      }

   }
   return bonus;
}
```

= goal) Line 8: { Line 9: bonus += 75; Line 10: } Line 11: else if (sales >= 0.8 * goal) Line 12: { Line 13: bonus += 50; Line 14: } Line 15 is blank. Line 16: } Line 17: return bonus; Line 18: } end code">

Part B – daily goal as constant

Select a point value to view scoring criteria, solutions, and/or examples and to score the response. +1 indicates a point earned and -1 indicates a general or question-specific penalty.

+1 [Skill 3.B] The response identifies properties of the new variable:

· int or "integer"

· final or "constant"

+1 [Skill 3.B] The response describes how the method changes:

· method calculateBonus would need to change so that it only has a single parameter numDays and would use the final variable to determine if the goal was met

Responses can still earn the point even if they omit final

Responses only earn the point if all described components are plausible, accurate, and consistent as related to the problem

0	1	2 ✓

Total number of points earned (minus penalties) is equal to 2.

☐ **+1** The response identifies properties of the new variable **(Points earned)**:
o int or "integer"
o final or "constant"

Quiz Review 1

☐ **+1** The response describes how the method changes **(Points earned)**:
o method `calculateBonus` would need to change so that it only has a single parameter `numDays` and would use the `final` variable to determine if the goal was met

Quiz Review 1

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

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.

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 nth digit of num.
       * Precondition: n >= 1 and n <= the number of digits in num
       */
     public static int getDigit(int num, int n)
     {
```

Quiz Review 1

```
        /* implementation not shown */
    }

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

Quiz Review 1

40. **(a)** Write 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
$(2x7) + (8x6) + (3x5) + (4x4) + (1x3) + (5x2) = 14 + 48 + 15 + 16 + 3 + 10 = 106.$
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 $(2x7) + (1x6) + (8x5) + (3x4) = 14 + 6 + 40 + 12 = 72.$
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 `n`th 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

Quiz Review 1

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)
```

Part A – getCheck method

Select a point value to view scoring criteria, solutions, and/or examples and to score the response. +1 indicates a point earned and -1 indicates a general or question-specific penalty.

+1 [Skill 3.A] Calls getNumberOfDigits and getDigit.

+1 [Skill 3.C] Calculates one partial sum = digit i * (8 - i)

+1 [Skill 3.C] Calculates all partial sums (*in the context of a loop, no bounds errors*).

+1 [Skill 3.C] Returns the last digit of the calculated sum as the check digit.

-1 (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)

-1 (x) Local variables used but none declared

Canonical Solution:

Quiz Review 1

```
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;
}
```

0	1	2	3	4

Total number of points earned (minus penalties) is equal to 4.

- ☐ **+1** Calls getNumberOfDigits and getDigit. **(Points earned)**
- ☐ **+1** Calculates one partial sum = digit i * (8 - i). **(Points earned)**
- ☐ **+1** Calculates all partial sums (*in the context of a loop, no bounds errors*). **(Points earned)**
- ☐ **+1** Returns the last digit of the calculated sum as the check digit. **(Points earned)**
- ☐ **-1** (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check) **(General penalties)**
- ☐ **-1** (x) Local variables used but none declared **(General penalties)**

Canonical Solution:

```
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;
}
```

Part B – isValid method

Select a point value to view scoring criteria, solutions, and/or examples and to score the response. +1 indicates a point earned and -1 indicates a general or question-specific penalty.

+1 [Skill 3.C] Obtains check digit of numWithCheckDigit.

+1 [Skill 3.C] Obtains number remaining in numWithCheckDigit after check digit removed.

+1 [Skill 3.A] Calls getCheck on number without check digit.

Quiz Review 1

+1 [Skill 3.C] Compares check digit of `numWithCheckDigit` and return value from `getCheck`

+1 [Skill 3.C] Returns `true` or `false` depending on the result of the previous comparison.

-1 (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)

-1 (x) Local variables used but none declared

Canonical Solution:

```
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;
   }
}
```

0	1	2	3	4	5

Total number of points earned (minus penalties) is equal to 5.

- ☐ **+1** Obtains check digit of `numWithCheckDigit`. **(Points earned)**
- ☐ **+1** Obtains number remaining in `numWithCheckDigit` after check digit removed. **(Points earned)**
- ☐ **+1** Calls `getCheck` on number without check digit. **(Points earned)**
- ☐ **+1** Compares check digit of `numWithCheckDigit` and return value from `getCheck`. **(Points earned)**
- ☐ **+1** Returns `true` or `false` depending on the result of the previous comparison. **(Points earned)**
- ☐ **-1** (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check) **(General penalties)**
- ☐ **-1** (x) Local variables used but none declared **(General penalties)**

Canonical Solution:

Quiz Review 1

```
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;
   }
}
```

Quiz Review 1

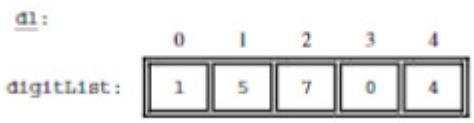
41. This question involves identifying and processing the digits of a non-negative integer. The declaration of the Digits class is shown below. You will write the constructor and one method for the Digits class.

```
public class Digits
{
   /** The list of digits from the number used to construct this object.
    *  The digits appear in the list in the same order in which they appear in the original number.
    */
   private ArrayList<Integer> digitList;

   /** Constructs a Digits object that represents num.
    *  Precondition: num >= 0
    */
   public Digits(int num)
   {  /* to be implemented in part (a) */  }

   /** Returns true if the digits in this Digits object are in strictly increasing order;
    *          false otherwise.
    */
   public boolean isStrictlyIncreasing()
   {  /* to be implemented in part (b) */  }
}
```

(a) Write the constructor for the Digits class. The constructor initializes and fills digitlist with the digits from the non-negative integer num. The elements in digitlist must be Integer objects representing single digits, and appear in the same order as the digits in num. Each of the following examples shows the declaration of a Digits object and the contents of digitist as initialized by the constructor.

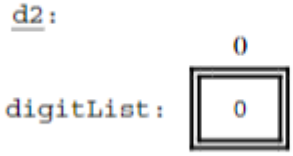
Example 1

```
Digits d1 = new Digits(15704);
```



Example 2

```
Digits d2 = new Digits(0);
```



Complete the Digits constructor below.

Quiz Review 1

```
/** Constructs a Digits object that represents num.
 *  Precondition: num >= 0
 */
public Digits(int num)
```

(b) Write the Digits method isStrictlyIncreasing. The method returns true if the elements of digitlist appear in strictly increasing order; otherwise, it returns false. A list is considered strictly increasing if each element after the first is greater than (but not equal to) the preceding element.

The following table shows the results of several calls to isStrictlyIncreasing.

Method call	Value returned
`new Digits(7).isStrictlyIncreasing()`	`true`
`new Digits(1356).isStrictlyIncreasing()`	`true`
`new Digits(1336).isStrictlyIncreasing()`	`false`
`new Digits(1536).isStrictlyIncreasing()`	`false`
`new Digits(65310).isStrictlyIncreasing()`	`false`

Complete method isStrictlyIncreasing below.

```
/** Returns true if the digits in this Digits object are in strictly increasing order;
 *          false otherwise.
 */
public boolean isStrictlyIncreasing()
```

Part (A) Digits constructor

Intent: *Initialize instance variable using passed parameter*

1 point is earned for: Constructs digitList

1 point is earned for: Identifies a digit innum

1 point is earned for: Adds at least one identified digit to a list

1 point is earned for: Adds all identified digits to a list *(must be in context of a loop)*

1 point is earned for: On exit: digitList contains all and only digits of num in the correct order

Quiz Review 1

```
public Digits(int num)
{
    digitList = new ArrayList<Integer>();
    if (num == 0)
    {
    digitList.add(new Integer(0));
    }
    while (num > 0)
    {
    digitList.add(0, new Integer(num % 10)); num /= 10;
    }
}
```

Question-Specific Penalties

-2 points for: (q) Uses confused identifier instead of digitList

0	1	2	3	4	5 ✓

The student response earns all of the following points:

Intent: *Initialize instance variable using passed parameter*

1 point is earned for: Constructs digitList

1 point is earned for: Identifies a digit innum

1 point is earned for: Adds at least one identified digit to a list

1 point is earned for: Adds all identified digits to a list *(must be in context of a loop)*

1 point is earned for: On exit: digitList contains all and only digits of num in the correct order

```
public Digits(int num)
{
    digitList = new ArrayList<Integer>();
```

Quiz Review 1

```
    if (num == 0)
    {
    digitList.add(new Integer(0));
    }
    while (num > 0)
    {
    digitList.add(0, new Integer(num % 10)); num /= 10;
    }
}
```

Question-Specific Penalties

-2 points for: (q) Uses confused identifier instead of digitList

Part (B) isStrictlyIncreasing

Intent: *Determine whether or not elements in digitList are in increasing order*

1 point is earned for: Compares at least one identified consecutive pair of digitList elements

1 point is earned for: Determines if a consecutive pair of digitList is out of order(must be in context of adigitList traversal)

1 point is earned for: Compares all necessary consecutive pairs of elements(no bounds errors)

1 point is earned for: Returns true iff all consecutive pairs of elements are in order; returns false otherwise

```
public boolean isStrictlyIncreasing()
{
    for (int i = 0; i < digitList.size()-1; i++)
    {
        if (digitList.get(i).intValue() >= digitList.get(i+1).intValue())
        {
        return false;
        }
    }
    return true;
```

Quiz Review 1

}

Question-Specific Penalties

-2 points for: (q) Uses confused identifier instead of digitList

0	1	2	3	4 ✓

The student response earns all of the following points:

Intent: *Determine whether or not elements in digitList are in increasing order*

1 point is earned for: Compares at least one identified consecutive pair of digitList elements

1 point is earned for: Determines if a consecutive pair of digitList is out of order(must be in context of adigitList traversal)

1 point is earned for: Compares all necessary consecutive pairs of elements(no bounds errors)

1 point is earned for: Returns true iff all consecutive pairs of elements are in order; returns false otherwise

```
public boolean isStrictlyIncreasing()
{
    for (int i = 0; i < digitList.size()-1; i++)
    {
        if (digitList.get(i).intValue() >= digitList.get(i+1).intValue())
        {
        return false;
        }
    }
    return true;
}
```

Question-Specific Penalties

-2 points for: (q) Uses confused identifier instead of digitList

Quiz Review 1

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

Notes:

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

4. This question involves generating a `String` that will be used as an identifier. You will write the `generateID` method of the following `Identifier` class.

```
public class Identifier
{
 /** Encodes a string as an integer and returns the encoded int value */
 public static int encodeToNumber(String str)
 {  /* implementation not shown */  }

 /** Returns an identifier string based on an input string, as described
 in part (a)
  *   Precondition: input is not null.
  */
 public static String generateID(String input)
 {  /* to be implemented in part (a) */  }

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

(a) Write the `generateID` method, which is used to transform an input string into a string that can be used as an identifier. The method creates and returns the identifier string based on the following rules.

- If the length of the input string is not divisible by 4, the method returns the string `"error"`.
- Every non-overlapping 4-character grouping of the input string is encoded as an integer using the helper method `encodeToNumber`. The sum of all the encoded values is calculated.
- If the sum is greater than 100, the method returns the original input string with `"3"` appended.
- Otherwise, the method returns the original input string with `"X"` appended.

The following table shows some examples of calls to the `generateID` method. Assume that all calls occur in the `Identifier` class.

Quiz Review 1

Call to `generateID`	**Possible Values Returned by** `encodeToNumber`	`generateID` **Return Value**
`generateID("treebook")`	`encodeToNumber("tree")` returns `17` `encodeToNumber("book")` returns `2`	`"treebookX"`

`generateID("doordesklion")`	`encodeToNumber("door")` returns `56`
	`encodeToNumber("desk")` returns `35`
	`encodeToNumber("lion")` returns `86`
		`"doordesklion3"`

Quiz Review 1

`generateID("today")`		`"error"` (because the length of `"today"` is not divisible by 4)

Complete method `generateID`. You must use `encodeToNumber` appropriately to receive full credit.

```
/** Returns an identifier string based on an input string, as described
in part (a)
 *  Precondition: input is not null.
 */
public static String generateID(String input)
```

(b) A programmer wants to modify the `Identifier` class to keep track of how many times a call to `generateID` returns `"error"`. The programmer would like to implement this change without making any changes to the signatures of `generateID` or `encodeToNumber` or overloading either method.

Write a description of how you would change the `Identifier` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

Part A – generateID

Select a point value to view scoring criteria, solutions, and/or examples and to score the response. +1 indicates a point earned and -1 indicates a general or question-specific penalty.

+1 [Skill 3.C] Uses length return value to determine divisible by 4

Responses can still earn the point even if they:

· determine the length of input incorrectly

· do not return

Quiz Review 1

"error"

or do not return any value in the case when the length of the input string is not a multiple of 4

Responses will not earn the point if they:

· incorrectly determine whether the length of input is divisible by 4

· use mod or MOD to test for divisibility

+1 [Skill 3.C] Loops over length of input string using character groupings (*no bounds errors*)

Responses can still earn the point even if they:

· inappropriately return inside the loop

· do not use 4-character groupings

Responses will not earn the point if they never build a group of multiple characters

+1 [Skill 3.A] Gets groups of 4 consecutive non-overlapping characters from input string

Responses can still earn the point even if they have a bounds error

Responses will not earn the point if they get incorrect substrings of 4 characters

+1 [Skill 3.A] Increments sum by result of call to encodeToNumber

Responses can still earn the point even if they:

· have a bounds error

· call encodeToNumber on something other than a 4-character substring

Responses will not earn the point if they:

· call encodeToNumber incorrectly

· compute sum incorrectly

+1 [Skill 3.C] Returns appropriate string based on length of input and sum of values in all three cases (algorithm point)

Responses can still earn the point even if they:

· incorrectly determine whether the length of input is divisible by 4

· use an incorrectly calculated sum

Responses will not earn the point if they return early

-1 (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)

-1 (x) Local variables used but none declared

Quiz Review 1

Canonical Solution:

```
public static String generateID(String input)
{
   if (input.length() % 4 != 0)
   {
      return "error";
   }

   int sum = 0;
   for (int i = 0; i < input.length(); i = i + 4)
   {
      String part = input.substring(i, i + 4);
      sum = sum + encodeToNumber(part);
   }

   if(sum > 100)
   {
      return input + "3";
   }
   else
   {
      return input + "X";
   }
}
```

100) Line 16: { Line 17: return input + "3"; Line 18: } Line 19: else Line 20: { Line 21: return input + "X"; Line 22: } Line 23: } end code">

0	1	2	3	4	5 ✓

Total number of points earned (minus penalties) is equal to 5.

- ☐ **+1** Uses length return value to determine divisible by 4 **(Points earned)**
- ☐ **+1** Loops over length of input string using character groupings (*no bounds errors*) **(Points earned)**
- ☐ **+1** Gets groups of 4 consecutive non-overlapping characters from input string **(Points earned)**
- ☐ **+1** Increments sum by result of call to encodeToNumber **(Points earned)**
- ☐ **+1** Returns appropriate string based on length of input and sum of values in all three cases (algorithm point) **(Points earned)**
- ☐ **-1** [penalty] (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check) **(General penalties)**
- ☐ **-1** [penalty] (x) Local variables used but none declared **(General penalties)**

Canonical Solution:

Quiz Review 1

```
public static String generateID(String input)
{
   if (input.length() % 4 != 0)
   {
      return "error";
   }

   int sum = 0;
   for (int i = 0; i < input.length(); i = i + 4)
   {
      String part = input.substring(i, i + 4);
      sum = sum + encodeToNumber(part);
   }

   if(sum > 100)
   {
      return input + "3";
   }
   else
   {
      return input + "X";
   }
}
```

100) Line 16: { Line 17: return input + "3"; Line 18: } Line 19: else Line 20: { Line 21: return input + "X"; Line 22: } Line 23: } end code">

Part B – Identifier class updates

Select a point value to view scoring criteria, solutions, and/or examples and to score the response. +1 indicates a point earned and -1 indicates a general or question-specific penalty.

+1 [Skill 3.B] The response identifies properties of the new variable:

· int or "integer"

· not local to a method

Responses can still earn the point even if they:

o specify the variable as an instance variable

o mention an integral data type (int , long ,etc.) or just mention "number"

Responses will not earn the point if they describe a change to the method header

+1 [Skill 3.B] The response describes how each identified variable or method would change or be implemented:

· generateID would need to change so that it increments the new variable when generateID returns "error"

Responses can still earn the point even if they describe a change to the method header

Responses only earn the point if all described components are plausible, accurate, and consistent as related to the problem

Quiz Review 1

0	1	2 ✓

Total number of points earned (minus penalties) is equal to 2.

- ☐ **+1** The response identifies properties of the new variable **(Points earned)**:
  o `int` or "integer"
  o not local to a method
- ☐ **+1** The response describes how each identified variable or method would change or be implemented **(Points earned)**:
  o `generateID` would need to change so that it increments the new variable when `generateID` returns `"error"`

Quiz Review 1

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

Notes:

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

2. This question involves validating a `String` that is being used as an identifier. You will write the `isValid` method of the following `Authenticate` class.

```
public class Authenticate
{
 /** Returns true if the input string has some property and returns
 false
  *  otherwise
  */
 public static boolean hasProperty(String str)
 {  /* implementation not shown */  }

 /** Returns true if the input string passes validation and returns
 false otherwise, as
  *   described in part (a)
  *  Precondition: input is not null.
  */
 public static boolean isValid(String input)
 {  /* to be implemented in part (a) */  }

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

(a) Write the `isValid` method, which validates an input string and returns a `boolean` value based on the following conditions. Every non-overlapping 3-character grouping of the input string is verified individually using the helper method `hasProperty`.

- If the length of the input string is not divisible by 3, the `isValid` method returns `false`.
- If the input string contains at least two 3-character groupings for which `hasProperty` returns `true`, the `isValid` method returns `true`.
- If there are fewer than two 3-character groupings for which `hasProperty` returns true, the `isValid` method returns `false`.

Quiz Review 1

The following table shows some examples of the intended behavior of `isValid`. Assume that all calls are made from within the `Authenticate` class.

Call to `isValid`	Possible Values Returned by `hasProperty`	`isValid` Return Value

`isValid("butterfly")`	`hasProperty("but")` returns `true`
	
	`hasProperty("ter")` returns `false`
	
	`hasProperty("fly")` returns `false`
		`false`

`isValid("turtle")`	`hasProperty("tur")` returns `true`
	
	`hasProperty("tle")` returns `true`
		`true`

`isValid("bear")`		`false` (because the length of `"bear"` is not divisible by 3)

You must use `hasProperty` appropriately to receive full credit.

Complete method `isValid`.

```
/** Returns true if the input string passes validation and returns false
otherwise, as
 *   described in part (a)
 *  Precondition: input is not null.
```

Quiz Review 1

```
 */
public static boolean isValid(String input)
```

(b) A programmer wants to modify the `Authenticate` class so that, in the `isValid` method, the rules for character groupings of 3 can vary between method calls. For example, in one call to `isValid`, the rules might check for divisibility by `4` with 4-character groupings, and in another call to `isValid`, the rule might check for divisibility by `5` with 5-character groupings. The programmer would like to implement this change without making any changes to the signature of the `isValid` method or overloading `isValid`.

Write a description of how you would change the `Authenticate` class in order to support this modification. **Do not write the program code for this change.**

Make sure to include the following in your response.

- Identify any new or modified variables or methods.
- Describe, for each new or revised variable or method, how it would change or be implemented, including visibility and type.

Part A – isValid

Select a point value to view scoring criteria, solutions, and/or examples and to score the response. +1 indicates a point earned and -1 indicates a general or question-specific penalty.

+1 [Skill 3.C] Uses length return value to determine divisible by 3

Responses can still earn the point even if they:

· determine the length of input incorrectly

· do not return false or do not return any value in the case when the length of the input string is not a multiple of 3

Responses will not earn the point if they:

· incorrectly determine whether the length of input is divisible by 3

· use mod or MOD to test for divisibility

+1 [Skill 3.C] Loops over length of input string using character groupings (*no bounds errors*)

Responses can still earn the point even if they:

· inappropriately return inside the loop

· do not use 3-character groupings

Responses will not earn the point if they never build a group of multiple characters

+1 [Skill 3.A] Gets groups of 3 consecutive non-overlapping characters from input string

Responses can still earn the point even if they have a bounds error

Responses will not earn the point if they get incorrect substrings of 3 characters

Quiz Review 1

+1 [Skill 3.C] Increments counter based on `hasProperty` return value

Responses can still earn the point even if they:

· have a bounds error

· call `hasProperty` on something other than a 3-character substring

Responses will not earn the point if they:

· call `hasProperty` incorrectly

· increment counter incorrectly

+1 [Skill 3.C] Returns appropriate `boolean` based on length of `input` and counter (algorithm point)

Responses can still earn the point even if they:

· incorrectly determine whether the length of `input` is divisible by 3

· increment counter incorrectly

Responses will not earn the point if they:

· return a value other than a `boolean`

· return early

-1 (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check)

-1 (x) Local variables used but none declared

Canonical Solution:

```
public static boolean isValid(String input)
{
   if (input.length() % 3 != 0)
   {
      return false;
   }

   int count = 0;

   for (int i = 0; i < input.length(); i += 3)
   {
      if (hasProperty(input.substring(i, i + 3)))
      {
         count++;
      }
   }
   return (count >= 2);
}
```

= 2); Line 18: } end code">

Quiz Review 1

0	1	2	3	4	5 ✓

Total number of points earned (minus penalties) is equal to 5.

- ☐ **+1** Uses `length` return value to determine divisible by 3 **(Points earned)**
- ☐ **+1** Loops over length of input string using character groupings (*no bounds errors*) **(Points earned)**
- ☐ **+1** Gets groups of 3 consecutive non-overlapping characters from input string **(Points earned)**
- ☐ **+1** Increments counter based on `hasProperty` return value **(Points earned)**
- ☐ **+1** Returns appropriate `boolean` based on length of `input` and counter (algorithm point) **(Points earned)**
- ☐ **-1** [penalty] (w) Extraneous code that causes side-effect (e.g., printing to output, incorrect precondition check) **(General penalties)**
- ☐ **-1** [penalty] (x) Local variables used but none declared **(General penalties)**

Canonical Solution:

```
public static boolean isValid(String input)
{
   if (input.length() % 3 != 0)
   {
      return false;
   }

   int count = 0;

   for (int i = 0; i < input.length(); i += 3)
   {
      if (hasProperty(input.substring(i, i + 3)))
      {
         count++;
      }
   }
   return (count >= 2);
}
```

= 2); Line 18: } end code">

Part B – Authenticate

Select a point value to view scoring criteria, solutions, and/or examples and to score the response. +1 indicates a point earned and -1 indicates a general or question-specific penalty.

+1 [Skill 3.B] The response identifies properties of the new variable:

· `int` or "integer"

Quiz Review 1

· not local to a method

Responses can still earn the point even if they:

o specify the variable as an instance variable

o describe a local variable that is assigned a value only once during the method call, where the value is generated randomly or from user input

o mention an integral data type (`int` , `long` ,etc.) or just mention "number"

Responses will not earn the point if they describe a change to the method header

+1 [Skill 3.B] The response describes how the method changes:

· `isValid` would need to change so that instead of checking for divisibility by 3 using % 3, it would use the new variable

· `isValid` would need to change so that instead of grouping characters by groups of 3, it would use the new variable

Responses can still earn the point even if they describe a change to the method header

Responses only earn the point if all described components are plausible, accurate, and consistent as related to the problem

0	1	2 ✓

Total number of points earned (minus penalties) is equal to 2.

- ☐ **+1** The response identifies properties of the new variable **(Points earned)**:
  o `int` or "integer"
  o not local to a method
- ☐ **+1** The response describes how the method changes **(Points earned)**:
  o `isValid` would need to change so that instead of checking for divisibility by 3 using % 3, it would use the new variable
  o `isValid` would need to change so that instead of grouping characters by groups of 3, it would use the new variable

Quiz Review 1

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

Notes:

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

A statistician is studying sequences of numbers obtained by repeatedly tossing a six-sided number cube. On each side of the number cube is a single number in the range of 1 to 6, inclusive, and no number is repeated on the cube. The statistician is particularly interested in runs of numbers. A run occurs when two or more consecutive tosses of the cube produce the same value. For example, in the following sequence of cube tosses, there are runs starting at positions 1, 6, 12, and 14

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Result	1	**5**	**5**	4	3	1	**2**	**2**	**2**	**2**	6	1	**3**	**3**	**5**	**5**	**5**	**5**

The number cube is represented by the following class.

```
public class NumberCube
{
  /** @return an integer value between 1 and 6, inclusive
   */
  public int toss()
  {  /* implementation not shown */  }

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

You will implement a method that collects the results of several tosses of a number cube and another method that calculates the longest run found in a sequence of tosses.

a. Write the method getCubeTosses that takes a number cube and a number of tosses as parameters. The method should return an array of the values produced by tossing the number cube the given number of times.

Complete method getCubeTosses below.

Quiz Review 1

```
/** Returns an array of the values obtained by tossing a number cube numTosses times.
 *  @param cube a NumberCube
 *  @param numTosses the number of tosses to be recorded
 *         Precondition: numTosses > 0
 *  @return an array of numTosses values
 */
public static int[] getCubeTosses(NumberCube cube, int numTosses)
```

b. Write the method getLongestRun that takes as its parameter an array of integer values representing a series of number cube tosses. The method returns the starting index in the array of a run of maximum size. A run is defined as the repeated occurrence of the same value in two or more consecutive positions in the array.

For example, the following array contains two runs of length 4, one starting at index 6 and another starting at index 14. The method may return either of those starting indexes.

If there are no runs of any value, the method returns -1.

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Result	1	5	5	4	3	1	2	2	2	2	6	1	3	3	5	5	5	5

Complete method getLongestRun below.

```
/** Returns the starting index of a longest run of two or more consecutive repeated values
 *  in the array values.
 *  @param values an array of integer values representing a series of number cube tosses
 *         Precondition: values.length > 0
 *  @return the starting index of a run of maximum size;
 *          -1 if there is no run
 */
public static int getLongestRun(int[] values)
```

Part A: getCubeTosses

4 points:

- **+1** constructs array
  - **+1/2** constructs an array of type int **or** size numTosses
  - **+1/2** constructs an array of type int **and** size numTosses
- **+2 1/2** processes tosses
  - **+1** repeats execution of statements numTosses times
  - **+1** tosses cube in context of iteration

Quiz Review 1

- **+1/2** collects results of tosses

- **+1/2** returns array of generated results

0	1	2	3	4 ✓

The student response earns four of the following points:

4 points:

- **+1** constructs array
  - **+1/2** constructs an array of type int **or** size numTosses
  - **+1/2** constructs an array of type int **and** size numTosses
- **+2 1/2** processes tosses
  - **+1** repeats execution of statements numTosses times
  - **+1** tosses cube in context of iteration
  - **+1/2** collects results of tosses
- **+1/2** returns array of generated results

Part B: getLongestRun

5 points:

- **+1** iterates over values
  - **+1/2** accesses element of values in context of iteration
  - **+1/2** accesses all elements of values, no out-of-bounds access potential
- **+1** determines existence of run of consecutive elements
  - **+1/2** comparison involving an element of values
  - **+1/2** comparison of consecutive elements of values
- **+1** always determines length of at least one run of consecutive elements
- **+1** identifies maximum length run based on all runs
- **+1** return value
  - +1/2 returns starting index of identified maximum length run

Quiz Review 1

- +1/2 returns -1 if no run identified

0	1	2	3	4	5 ✓

The student response earns five of the following points:

5 points:

- **+1** iterates over values
  - **+1/2** accesses element of values in context of iteration
  - **+1/2** accesses all elements of values, no out-of-bounds access potential
- **+1** determines existence of run of consecutive elements
  - **+1/2** comparison involving an element of values
  - **+1/2** comparison of consecutive elements of values
- **+1** always determines length of at least one run of consecutive elements
- **+1** identifies maximum length run based on all runs
- **+1** return value
  - +1/2 returns starting index of identified maximum length run
  - +1/2 returns -1 if no run identified