2021 AP Computer Science A International Exam

Updated September 2, 2024

Contents

1 Section I

The exam begins on the next 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[][] anArray = new int[10][8];
for (int j = 0; j < 8; j++)
{
   for (int k = 0; k < 10; k++)
   {
      anArray[j][k] = 5;
   }
}
```

The code segment causes an `ArrayIndexOutOfBoundsException` to be thrown. How many elements in `anArray` will be set to `5` before the exception is thrown?

(A) 0

(B) 8

(C) 9

(D) 64

(E) 80

2. Assume that `mat` has been declared as a 4×4 array of integers and has been initialized to contain all `1`s. Consider the following code segment.

```
int n = mat.length;
for (int j = 1; j < n; j++)
{
   for (int k = 1; k < n; k++)
   {
      mat[j][k] = mat[j - 1][k] + mat[j][k - 1];
   }
}
```

What is the value of `mat[2][2]` after the code segment has completed execution?

(A) 2

(B) 3

(C) 4

(D) 6

(E) 10

3. Consider the following class declarations.

```
public class Alpha
{
   private int answer()
   {
      return 10;
   }
}

public class Beta
{
   public double sample()
   {
      Alpha item = new Alpha();
      double temp = item.answer();
      return temp * 2.0;
   }
}
```

Which of the following best describes why an error occurs when the classes are compiled?

(A) The class `Alpha` does not have a defined constructor.

(B) The class `Alpha` must be declared as a subclass of `Beta`.

(C) The class `Beta` must be declared as a subclass of `Alpha`.

(D) The `answer` method cannot be accessed from a class other than `Alpha`.

(E) The result of the method call `item.answer()` cannot be assigned to a variable of type double.

4. Consider the following instance variable and incomplete method. The method is intended to return a string from the array `words` that would be last alphabetically.

```
private String[] words;
public String findLastWord()
{
   /* missing implementation */
}
```

Assume that `words` has been initialized with one or more strings containing only lowercase letters. Which of the following code segments can be used to replace */* missing implementation */* so that `findLastWord` will work as intended?

(A)

```
int maxIndex = 0;
for (int k = 0; k < words.length; k++)
{
   if (words[k].compareTo(maxIndex) > 0)
   {
      maxIndex = k;
   }
}
return words[maxIndex];
```

(B)

```
int maxIndex = 0;
for (int k = 1; k <= words.length; k++)
{
   if (words[k].compareTo(words[maxIndex]) > 0)
   {
      maxIndex = k;
   }
}
return words[maxIndex];
```

(C)

```
int maxIndex = 0;
for (int k = 1; k < words.length; k++)
{
   if (words[k].compareTo(words[maxIndex]) > 0)
   {
      maxIndex = k;
   }
}
return maxIndex;
```

(D)

```
String maxWord = words[0];
for (int k = 1; k < words.length; k++)
{
   if (words[k].compareTo(maxWord) > 0)
   {
      maxWord = k;
   }
}
return maxWord;
```

(E)

```
String maxWord = words[0];
for (int k = 1; k < words.length; k++)
{
   if (words[k].compareTo(maxWord) > 0)
   {
      maxWord = words[k];
   }
}
return maxWord;
```

5. Consider the following code segment.

```
ArrayList<String> colors = new ArrayList<String>();
colors.add("Red");
colors.add("Orange");
colors.set(1, "Yellow");
colors.add(1, "Green");
colors.set(colors.size() - 1, "Blue");
colors.remove(0);
System.out.println(colors);
```

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

(A) `[Red, Orange]`

(B) `[Red, Green]`

(C) `[Yellow, Blue]`

(D) `[Green, Blue]`

(E) `[Blue, Yellow]`

6. Consider the following class definition.

```
public class Box
{
   private double weight;
   /** Postcondition: weight is initialized to w. */
   public Box(double w)
   {
      /* implementation not shown */
   }
   public double getWeight()
   {
      return weight;
   }
   public void addWeight(double aw)
   {
      /* missing statement */
   }
}
```

The following code segment, which appears in a class other than `Box`, is intended to create a `Box` object `b1` with a weight of 2.2 units and then increase the weight of `b1` by 1.5 units.

```
Box b1 = new Box(2.2);
b1.addWeight(1.5);
```

Which of the following statements could replace */* missing statement */* so that the code segment works as intended?

(A) `aw += weight;`

(B) `aw += getWeight();`

(C) `weight += aw;`

(D) `weight += getWeight();`

(E) `return weight + aw;`

7. Consider the following three class declarations.

```
public class ClassOne
{
   public void methodA()
   { /* implementation not shown */ }
   public void methodB()
   { /* implementation not shown */ }
}
public class ClassTwo
{
   public void methodA()
   { /* implementation not shown */ }
}
public class ClassThree extends ClassOne
{
   public void methodB()
   { /* implementation not shown */ }
}
```

The following declarations occur in a method in another class.

```
ClassOne one = new ClassOne();
ClassTwo two = new ClassTwo();
ClassThree three = new ClassThree();
/* missing method call */
```

Which of the following replacements for */* missing method call */* will cause a compile-time error?

(A) `one.methodA();`

(B) `two.methodA();`

(C) `two.methodB();`

(D) `three.methodA();`

(E) `three.methodB();`

8. Consider the following class declaration.

```
public class Sample
{
   private int a;
   private double b;
   public Sample(int x, double y)
   {
      a = x;
      b = y;
   }
   // No other constructors
}
```

The following method appears in a class other than `Sample`.

```
public static void test()
{
   Sample object = new /* missing constructor call */ ;
}
```

Which of the following could be used to replace */* missing constructor call */* so that the method will compile without error?

(A) `Sample()`

(B) `Sample(int x = 10, double y = 6.2)`

(C) `Sample(int x, double y)`

(D) `Sample(10, 6.2)`

(E) `Sample(6.2, 10)`

9. 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

10. Consider the following method.

```
/** Precondition: Strings one and two have the same length. */
public static String combine(String one, String two)
{
   String res = "";
   for (int k = 0; k < one.length(); k++)
   {
      if (one.substring(k, k + 1).equals(two.substring(k, k + 1)))
      {
         res += one.substring(k, k + 1);
      }
      else
      {
         res += "0";
      }
   }
   return res;
}
```

What is returned as a result of the call `combine("10110", "01100")`?

(A) `"00000"`

(B) `"00100"`

(C) `"00101"`

(D) `"10110"`

(E) `"11011"`

11. 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

12. Consider the following incomplete method, which is intended to return `true` if the value of `y` is between the values of the other two parameters and `false` otherwise.

```
/** Precondition: x, y, and z have 3 different values. */
public static boolean compareThree(int x, int y, int z)
{
   return /* missing condition */ ;
}
```

The following table shows the results of several calls to `compareThree`.

Call	Result
`compareThree(4, 5, 6)`	`true`
`compareThree(6, 5, 4)`	`true`
`compareThree(5, 4, 6)`	`false`
`compareThree(3, 4, 4)`	violates precondition

Which of the following can be used to replace */* missing condition */* so that `compareThree` will work as intended when called with parameters that satisfy its precondition?

(A) `(x > y) && (x > z)`

(B) `(x > y) && (y > z)`

(C) `(x > y) || (y > z)`

(D) `(x > y) == (y > z)`

(E) `(x > y) != (y > z)`

Questions 13-14

Consider the following correct implementation of the insertion sort algorithm. The `insertionSort` method correctly sorts the elements of `ArrayList` data into increasing order.

```
public static void insertionSort(ArrayList<Integer> data)
{
   for (int j = 1; j < data.size(); j++)
   {
      int v = data.get(j);
      int k = j;
      {
         data.set(k, data.get(k - 1)); /* Statement 1 */
         k--;
      }
      data.set(k, v); /* Statement 2 */
      /* End of outer loop */
   }
}
```

13. Assume that `insertionSort` has been called with an `ArrayList` parameter that has been initialized with the following `Integer` objects.

    `[5, 2, 4, 1, 3, 6]`

    What will the contents of `data` be after three passes of the outside loop (i.e., when `j == 3` at the point indicated by */* End of outer loop */*)?

    (A) `[1, 2, 3, 4, 5, 6]`

    (B) `[1, 2, 3, 5, 4, 6]`

    (C) `[1, 2, 4, 5, 3, 6]`

    (D) `[2, 4, 5, 1, 3, 6]`

    (E) `[5, 2, 1, 3, 4, 6]`

14. Assume that `insertionSort` is called with an `ArrayList` parameter that has been initialized with the following `Integer` objects.

    `[1, 2, 3, 4, 5, 6]`

    How many times will the statements indicated by */* Statement 1 */* and */* Statement 2 */* execute?

	Statement 1	*Statement 2*
(A)	0	0
(B)	0	5
(C)	0	6
(D)	5	5
(E)	6	6

15. Consider the following method, which is intended to return the number of columns in the two-dimensional array `arr` for which the sum of the elements in the column is greater than the parameter `val`.

```
public int countCols(int[][] arr, int val)
{
   int count = 0;
   for (int col = 0; col < arr[0].length; col++) // Line 5
   {
      int sum = 0;
      for (int[] row : col) // Line 8
      {
         sum += row[col]; // Line 10
      }
      if (sum > val)
      {
         count++;
      }
   }
   return count;
}
```

The `countCols` method does not work as intended. Which of the following changes should be made so the method works as intended?

(A) Line 5 should be changed to `for (int col = 0; col < arr.length; col++)`

(B) Line 8 should be changed to `for (int row : col)`

(C) Line 8 should be changed to `for (int[] row : arr)`

(D) Line 10 should be changed to `sum += arr[col];`

(E) Line 10 should be changed to `sum += arr[row][col];`

16. Consider the following method, which is intended to count the number of times the letter `"A"` appears in the string `str`.

```
public static int countA(String str)
{
   int count = 0;
   while (str.length() > 0)
   {
      int pos = str.indexOf("A");
      if (pos >= 0)
      {
         count++;
         /* missing code */
      }
      else
      {
         return count;
      }
   }
   return count;
}
```

Which of the following should be used to replace */* missing code */* so that method `countA` will work as intended?

(A) `str = str.substring(0, pos);`

(B) `str = str.substring(0, pos + 1);`

(C) `str = str.substring(pos - 1);`

(D) `str = str.substring(pos);`

(E) `str = str.substring(pos + 1);`

17. Consider the following class declarations. Assume that each class has a no-argument constructor.

```
public class Food
{ /* implementation not shown */ }
public class Snack extends Food
{ /* implementation not shown */ }
public class Pizza extends Snack
{ /* implementation not shown */ }
```

Which of the following declarations will compile without error?

(A) `Food tacos = new Snack();`

(B) `Pizza cheesePizza = new Snack();`

(C) `Pizza sausagePizza = new Food();`

(D) `Snack pretzel = new Food();`

(E) `String Snack = new Pizza();`

18. Consider the following classes.

```
public class Ticket
{
   private double price;
   public Ticket(double p)
   {
      price = p;
   }
   public double getPrice()
   {
      return price;
   }
   public String toString()
   {
      return "Price is " + getPrice();
   }
}
public class DiscountTicket extends Ticket
{
   public DiscountTicket(double p)
   {
      super(p);
   }
   public double getPrice()
   {
      return super.getPrice() / 2.0;
   }
}
```

The following code segment appears in a class other than `Ticket` or `DiscountTicket`.

```
Ticket t = new DiscountTicket(10.0);
System.out.println(t);
```

What output, if any, is produced when the code segment is executed?

(A) `5.0`

(B) `10.0`

(C) `Price is 5.0`

(D) `Price is 10.0`

(E) There is no output because the code does not compile.

19. 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)`

20. Consider the following method.

```
public String exercise(int input)
{
   if (input < 10)
   {
      return "alpha";
   }
   if (input < 5)
   {
      return "beta";
   }
   if (input < 1)
   {
      return "gamma";
   }
   return "delta";
}
```

Assume that the `int` variable `x` has been initialized in another method in the same class. Which of the following describes the conditions under which the method call `exercise(x)` will return `"gamma"`?

(A) The method will never return `"gamma"`.

(B) The method will return `"gamma"` if `x` is less than `1`.

(C) The method will return `"gamma"` if `x` is between `1` and `4`, inclusive.

(D) The method will return `"gamma"` if `x` is between `5` and `9`, inclusive.

(E) The method will always return `"gamma"`.

21. Consider the following method.

```
public static int what(String str, String check)
{
   int num = -1;
   int len = check.length();
   for (int k = 0; k + len <= str.length(); k++)
   {
      String a = str.substring(k, k + len);
      if (a.equals(check))
      {
         num = k;
      }
   }
   return num;
}
```

Assume that `check` occurs at least once in `str`. Which of the following best describes the value returned by the `what` method?

(A) The number of times the string `check` occurs in `str`

(B) The index of the first occurrence of `check` inside `str`

(C) The index of the last occurrence of `check` inside `str`

(D) The number of substrings in `str` with the same length as `check`

(E) The number of substrings in `str` that do not match `check`

22. Assume that `x` and `y` are variables of type `int`. Which of the following Java expressions never results in a division by zero?

(A) `(y / x) == 0`

(B) `((y / x) == 0) && (x != 0)`

(C) `((y / x) == 0) || (x != 0)`

(D) `(x != 0) && ((y / x) == 0)`

(E) `(x != 0) || ((y / x) == 0)`

23. Consider the following code segment.

```
for (int num = 0; num < 10; num += 2)
{
   for (int val = 0; val < 5; val++)
   {
      System.out.println("hop");
} }
```

How many times will `System.out.println("hop")` be executed?

(A) 0

(B) 5

(C) 10

(D) 25

(E) 50

24. 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`

25. Consider the following method.

```
public static int mystery(boolean a, boolean b, boolean c)
{
   int answer = 7;
   if (!a)
   {
      answer += 1;
   }
   if (b)
   {
      answer += 2;
   }
   if (c)
   {
      answer += 4;
   }
   return answer;
}
```

Which of the following method calls will return the value `11` ?

(A) `mystery(true, true, true)`

(B) `mystery(true, false, true)`

(C) `mystery(false, true, false)`

(D) `mystery(false, false, true)`

(E) `mystery(false, false, false)`

26. Consider the following method.

```
public static int mystery(int value)
{
   int sum = 0;
   int[] arr = {1, 4, 2, 5, 10, 3, 6, 4};
   for (int item : arr)
   {
      if (item > value)
      {
         sum += item;
      }
   }
   return sum;
}
```

What value is returned as a result of the call `mystery(4)` ?

(A) `6`

(B) `15`

(C) `21`

(D) `29`

(E) `35`

27. Consider the following method.

```
public static int mystery(int n)
{
   if (n <= 0)
   {
      return 0;
   }
   else
   {
      return n + mystery(n - 2);
   }
}
```

What value is returned as a result of the call `mystery(9)` ?

(A) `0`

(B) `9`

(C) `16`

(D) `24`

(E) `25`

28. Consider the following method.

```
public static int mystery(int[] arr)
{
   int count = 0;
   int curr = arr[arr.length - 1];
   for (int value : arr)
   {
      if (value > curr)
      {
         count = count + 1;
      }
      else
      {
         count = count - 1;
      }
      curr = value;
   }
   return count;
}
```

The following code segment appears in another method of the same class.

```
int[] arr = {4, 14, 15, 3, 14, 18, 19};
System.out.println(mystery(arr));
```

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

(A) -7

(B) -6

(C) 3

(D) 5

(E) 7

29. Consider the following code segment.

```
int[] numbers = new int[5];
numbers[0] = 2;
numbers[1] = numbers[0] + 1;
numbers[numbers[0]] = numbers[1];
for (int x = 3; x < numbers.length; x++)
{
   numbers[x] = numbers[x - 1] * 2;
}
```

Which of the following represents the contents of the array `numbers` after the code segment is executed?

(A) {2, 3, 0, 0, 0}

(B) {2, 3, 1, 2, 4}

(C) {2, 3, 3, 6, 9}

(D) {2, 3, 3, 6, 12}

(E) {2, 4, 8, 16, 32}

30. Consider the following attempts at method overloading.

I.

```
public class Overload
{
   public int average(int x, int y)
   { /* implementation not shown */ }
   public int average(int value1, int value2)
   { /* implementation not shown */ }
   // There may be instance variables, constructors,
   // and methods that are not shown.
}
```

II.

```
public class Overload
{
   public int average(int x, int y)
   { /* implementation not shown */ }
   public int average(int x, int y, int z)
   { /* implementation not shown */ }
   // There may be instance variables, constructors
   // and methods that are not shown.
}
```

III.

```
public class Overload
{
   public int average(int x, int y)
   { /* implementation not shown */ }
   public int average(double x, double y)
   { /* implementation not shown */ }
   // There may be instance variables, constructors,
   // and methods that are not shown.
}
```

Which of the attempts at method overloading will compile without error?

(A) I only

(B) II only

(C) III only

(D) II and III only

(E) I, II, and III

31. Consider the following method.

```
public static void mystery(int[] a, int index)
{
   if (index < a.length)
   {
      mystery(a, index + 1);
   }
   if (index > 0)
   {
      System.out.print(a[index - 1]);
   }
}
```

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

```
int[] array = {6, 8, 7, 9, 5};
mystery(array, 0);
```

(A) 5978

(B) 8795

(C) 59786

(D) 68795

(E) Many digits are printed because of infinite recursion.

32. 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

33. Consider the following code segment.

```
for (int k = 0; k < 9; k = k + 2)
{
   if ((k % 2) != 0)
   {
      System.out.print(k + " ");
   }
}
```

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

(A) `0 2 4 6 8 10`

(B) `0 2 4 6 8`

(C) `1 3 5 7 9`

(D) `1 3 5 7`

(E) Nothing is printed.

34. Consider the following method.

```
public static void whatIsIt(int a, int b)
{
   if ((a < b) && (a > 0))
   {
      System.out.println("W");
   }
   else if (a == b)
   {
      if (b > 0)
      {
         System.out.println("X");
      }
      else if (b < 0)
      {
         System.out.println("Y");
      }
      {
         System.out.println("Z");
      }
   }
}
```

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

I. `whatIsIt(10, 10)`

II. `whatIsIt(-5, 5)`

III. `whatIsIt(7, 10)`

(A) I only

(B) II only

(C) III only

(D) I and III

(E) II and III

35. Consider the following method, which returns an `int` based on its parameter `x`.

```
public static int puzzle(int x)
{
   if (x > 20)
   {
      x -= 2;
   }
   else if (x % 2 == 0) // Line 7
   {
      x += 4;
   }
   return x;
}
```

Consider a modification to the method that eliminates the `else` from line 7 so that line 7 becomes

```
if (x % 2 == 0) // Modified line 7
```

For which of the following values of `x` would the return values of the original method and the modified method differ?

(A) `0`

(B) `5`

(C) `14`

(D) `22`

(E) `25`

36. Consider the following recursive method.

```
/** Precondition: 0 <= numVals <= nums.length */
public static int mystery(int[] nums, int v, int numVals)
{
   if (numVals == 0)
   {
      return 0;
   }
   else if (v == nums[numVals - 1])
   {
      return 1 + mystery(nums, v, numVals - 1);
   }
   else
   {
      return mystery(nums, v, numVals - 1);
   }
}
```

Which of the following best describes the value returned by the call `mystery(nums, v, nums.length)` ?

(A) The value `0` is returned.

(B) The value `1` is returned.

(C) The number of times that `v` occurs in `nums` is returned.

(D) The number of times that `numVals` occurs in `nums` is returned.

(E) Nothing is returned. A runtime error occurs because of infinite recursion.

37. Consider the following class declarations.

```
public class Shoe
{
   private String shoeBrand;
   private String shoeModel;
   public Shoe(String brand, String model)
   {
      shoeBrand = brand;
      shoeModel = model;
   }
   // No other constructors
}
public class Boot extends Shoe
{
   private double heelHeight;
   public Boot(String brand, String model, double height)
   {
      /* missing implementation */
   }
}
```

Which of the following should be used to replace /* *missing implementation* */ so that all instance variables are initialized with parameters?

(A)
```
shoeBrand = brand;
shoeModel = model;
heelHeight = height;
```

(B)
```
super();
heelHeight = height;
```

(C)
```
super(brand, model);
```

(D)
```
heelHeight = height;
super(brand, model);
```

(E)
```
super(brand, model);
heelHeight = height;
```

38. Consider the following incomplete method, which is intended to return the sum of all the elements in its array parameter.

```
/** Precondition: data.length > 0 */
public static int total(int[] data)
{
   int result = 0;
   /* missing code */
   return result;
}
```

The following code segments have been proposed to replace /* *missing code* */.

I.

```
for (int k = 0; k < data.length; k++)
{
   result += k;
}
```

II.

```
for (int d : data)
{
   result += d;
}
```

III.

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

Which of the replacements for /* *missing code* */ could be used so that `total` will work as intended?

(A) I only

(B) II only

(C) III only

(D) I and II

(E) II and III

39. Consider the following method.

```
public static void methodX(int[] values)
{
   for (int j = 0; j < values.length - 1; j++)
   {
      for (int k = j + 1; k < values.length; k++)
      {
         if (values[k] < values[j])
         {
            values[j] = values[k];
         }
      }
   }
}
```

The following code segment appears in another method of the same class.

```
int[] numbers = {45, 1, 56, 10};
methodX(numbers);
```

Which of the following represents the contents of the array `numbers` after the code segment is executed?

(A) `{1, 1, 10, 10}`

(B) `{1, 10, 45, 56}`

(C) `{45, 1, 56, 10}`

(D) `{45, 45, 56, 45}`

(E) `{56, 45, 10, 1}`

40. Consider the following code segment.

```
int num = 1;
for (int k = 2; k < 10; k++)
{
   num = 0;
   num = num + k;
}
```

What will be the value of `num` after the loop is executed?

(A) `2`

(B) `9`

(C) `10`

(D) `44`

(E) `45`

STOP
END OF SECTION I

2 Section II

Section II begins on the next page.

COMPUTER SCIENCE A

Section II

Time—1 hour and 30 minutes

4 Questions

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

Notes:

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

1. A manufacturer wants to keep track of the average of the ratings that have been submitted for an item using a running average. The algorithm for calculating a running average differs from the standard algorithm for calculating an average, as described in part (a).

   A partial declaration of the `RunningAverage` class is shown below. You will write two methods of the `RunningAverage` class.

```
public class RunningAverage
{
   /** The number of ratings included in the running average. */
   private int count;
   /** The average of the ratings that have been entered. */
   private double average;
   // There are no other instance variables.
   /** Creates a RunningAverage object.
   * Postcondition: count is initialized to 0 and average is
   * initialized to 0.0.
   */
   public RunningAverage()
   { /* implementation not shown */ }
   /** Updates the running average to reflect the entry of a new
   * rating, as described in part (a).
   */
   public void updateAverage(double newVal)
   { /* to be implemented in part (a) */ }
   /** Processes num new ratings by considering them for inclusion
   * in the running average and updating the running average as
   * necessary. Returns an integer that represents the number of
   * invalid ratings, as described in part (b).
   * Precondition: num > 0
   */
   public int processNewRatings(int num)
   { /* to be implemented in part (b) */ }
   /** Returns a single numeric rating. */
   public double getNewRating()
   { /* implementation not shown */ }
}
```

   (a) Write the method `updateAverage`, which updates the `RunningAverage` object to include a new rating. To update a running average, add the new rating to a calculated total, which is the number of ratings times the current running average. Divide the new total by the incremented count to obtain the new running average.

For example, if there are 4 ratings with a current running average of 3.5, the calculated total is 4 times 3.5, or 14.0. When a fifth rating with a value of 6.0 is included, the new total becomes 20.0. The new running average is 20.0 divided by 5, or 4.0.

Complete method `updateAverage`.

```
/** Updates the running average to reflect the entry of a new
* rating, as described in part (a).
*/
public void updateAverage(double newVal)
```

(b) Write the `processNewRatings` method, which considers `num` new ratings for inclusion in the running average. A helper method, `getNewRating`, which returns a single rating, has been provided for you.

The running average must only be updated with ratings that are greater than or equal to zero. Ratings that are less than 0 are considered invalid and are not included in the running average.

The `processNewRatings` method returns the number of invalid ratings. See the table below for three examples of how calls to `processNewRatings` should work.

Statement	**RatingsGenerated**	`processNewRatings` **Return Value**	**Comments**
`processNewRatings(2)`	`2.5, 4.5`	`0`	Both new ratings are included in the running average.
`processNewRatings(1)`	`-2.0`	`1`	No new ratings are included in the running average.
`processNewRatings(4)`	`0.0, -2.2,`	`2`	Two new ratings (`0.0` and `3.5`) are included in the running average.

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

```
/** Processes num new ratings by considering them for inclusion
* in the running average and updating the running average as
* necessary. Returns an integer that represents the number of
* invalid ratings, as described in part (b).
* Precondition: num > 0
*/
public int processNewRatings(int num)
```

2. The `Bus` class simulates the activity of a bus. A bus moves back and forth along a single route, making stops along the way. The stops on the route are numbered consecutively starting from 1 up to and including a number that is provided when the `Bus` object is created. You may assume that the number of stops will always be greater than 1.

   The bus starts at the first stop and is initially heading toward the last stop. At each step of the simulation, the bus is at a particular stop and is heading toward either the first or last stop. When the bus reaches the first or last stop, it reverses direction. The following table contains a sample code execution sequence and the corresponding results.

Statement or Expression	**Value returned(blank if novalue)**	**Comment**
`Bus bus1 = new Bus(3);`		The route for `bus1` has three stops numbered 1-3.
`bus1.getCurrentStop();`	`1`	`bus1` is at stop 1 (first stop on the route).
`bus1.move();`		`bus1` moves to the next stop (2).
`bus1.getCurrentStop();`	`2`	`bus1` is at stop 2.
`bus1.move();`		`bus1` moves to the next stop (3).
bus1.getCurrentStop();	`3`	`bus1` is at stop 3.
`bus1.move();`		`bus1` moves to the next stop (2).
`bus1.getCurrentStop();`	`2`	`bus1` is at stop 2.
`bus1.move();`		`bus1` moves to the next stop (1).
`bus1.move();`		`bus1` moves to the next stop (2).
`bus1.getCurrentStop();`	`2`	`bus1` is at stop 2.
`bus1.getCurrentStop();`	`2`	`bus1` is still at stop 2.
`Bus bus2 = new Bus(5);`		The route for `bus2` has five stops numbered 1-5.
`bus1.getCurrentStop();`	`2`	`bus1` is still at stop 2.
`bus2.getCurrentStop();`	`1`	`bus2` is at stop 1 (first stop on the route).

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

3. This question involves generating a list of possible nicknames from a person's name. For the purposes of this question, a person's name contains a first (given) name and a last (family) name. You may assume that all punctuation has been removed so that names contain only letters and that all letters are in uppercase.

   A partial declaration of the `NicknameGenerator` class is shown below. You will write two methods of the `NicknameGenerator` class.

```
public class NicknameGenerator
{
   /** The person's first name in all uppercase letters, initialized
   * by the constructor.
   */
   private String firstName;
   /** The person's last name in all uppercase letters, initialized
   * by the constructor.
   */
   private String lastName;
   // Constructor not shown
   /** Returns the number of vowels in lastName. */
   private int numVowels()
   { /* implementation not shown */ }
   /** Returns the index of the first vowel in lastName.
   * Returns -1 if there are no vowels in lastName.
   */
   private int indexOfFirstVowel()
   { /* implementation not shown */ }
   /** Returns a list of shortened last names, as described
   * in part (a).
   */
   public ArrayList<String> shortLastNames()
   { /* to be implemented in part (a) */ }
   /** Returns a list of nicknames, as described in part (b). */
   public ArrayList<String> nicknames()
   { /* to be implemented in part (b) */ }
}
```

   (a) Write the method `shortLastNames`, which returns a list of the shortened forms of a last name, according to the rules below. Each shortened last name must appear exactly once in the list.

   The following rules are used to produce the list of shortened last names.

   - If the last name contains fewer than two vowels, the list contains the complete last name as its only element.
   - If the last name contains two or more vowels, the first shortened last name in the list contains all characters from the original last name up to and including the first vowel. Each subsequent shortened last name is produced by adding one additional character from the original last name. The list ends with the complete last name.

   The `NicknameGenerator` class provides two helper methods. The method `numVowels` returns the number of vowels in `lastName`. The method `indexOfFirstVowel` returns the index of the first vowel in `lastName` or `-1` if there are no vowels in `lastName`.

In the table below, the vowels in each last name are underlined.

Last name	List of shortened last names
NG	`["NG"]`
SMITH	`["SMITH"]`
LOPES	`["LO", "LOP", "LOPE", "LOPES"]`
ASHE	`["A", "AS", "ASH", "ASHE"]`

Complete method `shortLastNames`. The helper methods `numVowels` and `indexOfFirstVowel` have been provided for you.

```
/** Returns a list of shortened last names, as described
* in part (a).
*/
public ArrayList<String> shortLastNames()
```

(b) Write the method `nicknames`, which returns a list of all possible nicknames. Each nickname must appear exactly once in the list.

A nickname is generated with the first letter of the first name, followed by `"-"`, followed by one of the possible shortened forms of the last name. The table below shows several names and the list of nicknames that are generated from the name.

Name	List of nicknames
CELESTE NG	`["C-NG"]`
GLEN SMITH	`["G-SMITH"]`
JUANITA LOPES	`["J-LO", "J-LOP", "J-LOPE", "J-LOPES"]`
MARY ASHE	`["M-A", "M-AS", "M-ASH", "M-ASHE"]`

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

Complete method `nicknames`.

```
/** Returns a list of nicknames, as described in part (b). */
public ArrayList<String> nicknames()
```

4. This question involves performing arithmetic operations on two-dimensional (2D) arrays of integers. You will write two static methods, both of which are in a class named `MatrixOp` (not shown).

(a) Write the method `diagonalOp`, which returns the sum of the products of the corresponding entries on the main diagonals of two given square 2D arrays that have the same dimensions. The main diagonal goes from the top-left corner to the bottom-right corner in a square 2D array.

For example, assume that `mat1` and `mat2` are properly defined 2D arrays containing the values shown below. The main diagonals have been shaded in gray.

`mat1`

2	4	2
8	**5**	1
4	2	**4**

`mat2`

-1	8	9
6	**3**	5
5	1	**2**

After the call `int sum = MatrixOp.diagonalOp(mat1, mat2);`, `sum` would contain `21`, as illustrated below.

```
sum = (2 * -1) + (5 * 3) + (4 * 2) = 21
```

Complete method `diagonalOp`.

```
/** Returns an integer, as described in part (a).
* Precondition: matA and matB are 2D arrays that are both square,
* have at least one row, and have the same dimensions.
*/
public static int diagonalOp(int[][] matA, int[][] matB)
```

(b) Write the method `expandMatrix`, which returns an expanded version of a given 2D array. To expand a 2D array, a new 2D array must be created and filled with values such that each element of the original 2D array occurs a total of four times in the new 2D array, arranged as shown in the example below.

For example, assume that `mat3` is a properly defined 2D array containing the values shown below.

mat3

	0	1	2
0	-1	2	3
1	5	4	6

After the call `int[][] mat4 = MatrixOp.expandMatrix(mat3)`, the array `mat4` would contain the values shown below.

mat4

	0	1	2	3	4	5
0	-1	-1	2	2	3	3
1	-1	-1	2	2	3	3
2	5	5	4	4	6	6
3	5	5	4	4	6	6

Complete method `expandMatrix`.

```
/** Returns a 2D array, as described in part (b).
* Precondition: matA is a 2D array with at least one row and
* at least one column.
*/
public static int[][] expandMatrix(int[][] matA)
```

STOP
END OF EXAM