

chap 6: Array

1. Consider the following method.

```
public static void addOneToEverything(int[] numbers)
{
    for (int j = 0; j < numbers.length; j++)
    {
        numbers[j]++;
    }
}
```

Which of the following code segments, if any, can be used to replace the body of the method so that `numbers` will contain the same values?

```
for (int num : numbers)
{
    num++;
}
```

I.

```
for (int num : numbers)
{
    num[j]++;
}
```

II.

```
for (int num : numbers)
{
    numbers[num]++;
}
```

III.

- (A) I only
(B) I and III only
(C) II and III only
(D) I, II, and III
(E) None of the code segments will return an equivalent result.

chap 6: Array

2. Consider the following method. Method `allEven` is intended to return `true` if all elements in array `arr` are even numbers; otherwise, it should return `false`.

```
public boolean allEven(int[] arr)
{
    boolean isEven = /* expression */ ;

    for (int k = 0; k < arr.length; k++)
    {
        /* loop body */
    }

    return isEven;
}
```

Which of the following replacements for */* expression */* and */* loop body */* should be used so that method `allEven` will work as intended?

chap 6: Array

(A)

<u><i>/* expression */</i></u>	<u><i>/* loop body */</i></u>
false	if ((arr[k] % 2) == 0) isEven = true;

(B)

<u><i>/* expression */</i></u>	<u><i>/* loop body */</i></u>
false	if ((arr[k] % 2) != 0) isEven = false; else isEven = true;

(C)

<u><i>/* expression */</i></u>	<u><i>/* loop body */</i></u>
true	if ((arr[k] % 2) != 0) isEven = false;

chap 6: Array

(D)

<i><u>/* expression */</u></i>	<i><u>/* loop body */</u></i>
true	<pre>if ((arr[k] % 2) != 0) isEven = false; else isEven = true;</pre>

(E)

<i><u>/* expression */</u></i>	<i><u>/* loop body */</u></i>
true	<pre>if ((arr[k] % 2) == 0) isEven = false; else isEven = true;</pre>

chap 6: Array

3. 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?

chap 6: Array

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

**chap 6: Array**

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

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

The code segment does not work as intended. Which of the following changes should be made so the code segment works as intended?

- (A) The Boolean expression in the `for` loop header should be changed to `i < animals.length`.
  - (B) The Boolean expression in the `for` loop header should be changed to `i < animals.length - 1`.
  - (C) The Boolean expression in the `for` loop header should be changed to `i < animals[i].length`.
  - (D) The condition in the `if` statement should be changed to `animals[i].equals(str)`.
  - (E) The condition in the `if` statement should be changed to `animals[i].substring(str)`.
5. Consider the following incomplete method that is intended to return an array that contains the contents of its first array parameter followed by the contents of its second array parameter.

```
public static int[] append(int[] a1, int[] a2)
{
 int[] result = new int[a1.length + a2.length];

 for (int j = 0; j < a1.length; j++)
 result[j] = a1[j];

 for (int k = 0; k < a2.length; k++)
 result[/* index */] = a2[k];

 return result;
}
```

Which of the following expressions can be used to replace `/* index */` so that `append` will work as intended?

- (A) `j`
- (B) `k`
- (C) `k + a1.length - 1`
- (D) `k + a1.length`
- (E) `k + a1.length + 1`

**chap 6: Array**

6. Consider the following instance variable and incomplete method. The method `calcTotal` is intended to return the sum of all values in `vals`.

```
private int[] vals;

public int calcTotal()
{
 int total = 0;

 /* missing code */

 return total;
}
```

Which of the code segments shown below can be used to replace */\* missing code \*/* so that `calcTotal` will work as intended?

- I. `for (int pos = 0; pos < vals.length; pos++)`

```
{
 total += vals[pos];
}
```

- II. `for (int pos = vals.length; pos > 0; pos--)`

```
{
 total += vals[pos];
}
```



**chap 6: Array**

```
III. int pos = 0;

while (pos < vals.length)

{

 total += vals[pos];

 pos++;
```

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

7. Consider the following instance variable and method.

```
private int[] array;

/** Precondition: array.length > 0
 */
public int checkArray()
{
 int loc = array.length / 2;
 for (int k = 0; k < array.length; k++)
 {
 if (array[k] > array[loc])
 {
 loc = k;
 }
 }
 return loc;
}
```

Which of the following is the best postcondition for checkArray ?

**chap 6: Array**

- (A) Returns the index of the first element in array array whose value is greater than array [loc]
- (B) Returns the index of the last element in array array whose value is greater than array [loc]
- (C) Returns the largest value in array array
- (D) Returns the index of the largest value in array array
- (E) Returns the index of the largest value in the second half of array array

**chap 6: Array**

8. Consider the following incomplete method that is intended to return a string formed by concatenating elements from the parameter words. The elements to be concatenated start with startIndex and continue through the last element of words and should appear in reverse order in the resulting string.

```
/** Precondition: words.length > 0;
 * startIndex >= 0
 */
public static String concatWords(String[] words, int startIndex)
{
 String result = "";

 /* missing code */

 return result;
}
```

For example, the following code segment uses a call to the concatWords method.

```
String[] things = {"Bear", "Apple", "Gorilla", "House", "Car"};
System.out.println(concatWords(things, 2));
```

When the code segment is executed, the string "CarHouseGorilla" is printed.

The following three code segments have been proposed as replacements for */\* missing code \*/*.

- I.     

```
for (int k = startIndex; k < words.length; k++)
{
 result += words[k] + words[words.length - k - 1];
}
```
- II.     

```
int k = words.length - 1;
while (k >= startIndex)
{
 result += words[k];
 k--;
}
```
- III.    

```
String[] temp = new String[words.length];
for (int k = 0; k <= words.length / 2; k++)
{
 temp[k] = words[words.length - k - 1];
 temp[words.length - k - 1] = words[k];
}

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

Which of these code segments can be used to replace */\* missing code \*/* so that concatWords will work as intended?

**chap 6: Array**

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

9. 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

**chap 6: Array**

10. The method `countTarget` below is intended to return the number of times the value `target` appears in the array `arr`. The method may not work as intended.

```
public int countTarget(int[] arr, int target)
{
 int count = 0;
 for (int j = 0; j <= arr.length; j++) // line 4
 {
 if (arr[j] == target)
 {
 count++;
 }
 }
 return count;
}
```

Which of the following changes, if any, can be made to line 4 so that the method will work as intended?

- (A) Changing `int j = 0;` to `int j = 1;`
  - (B) Changing `j <= arr.length;` to `j < arr.length;`
  - (C) Changing `j <= arr.length;` to `j < arr.length - 1;`
  - (D) Changing `j <= arr.length;` to `j < arr.length + 1;`
  - (E) No change is necessary; the method works correctly as is.
11. Consider the following code segment, which is intended to print the sum of all elements of an array.

```
int[] arr = {10, 5, 1, 20, 6, 25};
int sum = 0;
for (int k = 0; k <= arr.length; k++)
{
 sum += arr[k];
}
System.out.println("The sum is " + sum);
```

A runtime error occurs when the code segment is executed. Which of the following changes should be made so that the code segment works as intended?

- (A) The `for` loop header should be replaced with `for (int k = 0; k < arr.length; k++)`.
- (B) The `for` loop header should be replaced with `for (int k = 0; k <= arr.length; k--)`.
- (C) The `for` loop header should be replaced with `for (int k = 1; k <= arr.length - 1; k++)`.
- (D) The statement in the body of the `for` loop should be replaced with `sum += arr[0]`.
- (E) The statement in the body of the `for` loop should be replaced with `sum += arr[k - 1]`.

**chap 6: Array**

12. Consider the problem of finding the maximum value in an array of integers. The following code segments are proposed solutions to the problem. Assume that the variable `arr` has been defined as an array of `int` values and has been initialized with one or more values.

```
I. int max = Integer.MIN_VALUE;
 for (int value : arr)
 {
 if (max < value)
 {
 max = value;
 }
 }
```

```
II. int max = 0;
 boolean first = true;
 for (int value : arr)
 {
 if (first)
 {
 max = value;
 first = false;
 }
 else if (max < value)
 {
 max = value;
 }
 }
```

```
III. int max = arr[0];
 for (int k = 1; k < arr.length; k++)
 {
 if (max < arr[k])
 {
 max = arr[k];
 }
 }
```

Which of the code segments will always correctly assign the maximum element of the array to the variable `max` ?

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

**chap 6: Array**

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

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

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

## chap 6: Array

(A)

|                                  |                                      |
|----------------------------------|--------------------------------------|
| <i>/* missing condition 1 */</i> | <i>/* missing condition 2 */</i>     |
| <code>arr[j] == arr[k]</code>    | <code>valCount &gt; modeCount</code> |

(B)

|                                  |                                      |
|----------------------------------|--------------------------------------|
| <i>/* missing condition 1 */</i> | <i>/* missing condition 2 */</i>     |
| <code>arr[j] == arr[k]</code>    | <code>modeCount &gt; valCount</code> |

(C)

|                                  |                                      |
|----------------------------------|--------------------------------------|
| <i>/* missing condition 1 */</i> | <i>/* missing condition 2 */</i>     |
| <code>arr[j] != arr[k]</code>    | <code>valCount &gt; modeCount</code> |

(D)

|                                  |                                      |
|----------------------------------|--------------------------------------|
| <i>/* missing condition 1 */</i> | <i>/* missing condition 2 */</i>     |
| <code>arr[j] != arr[k]</code>    | <code>modeCount &gt; valCount</code> |

(E)

|                                  |                                    |
|----------------------------------|------------------------------------|
| <i>/* missing condition 1 */</i> | <i>/* missing condition 2 */</i>   |
| <code>arr[j] != arr[k]</code>    | <code>modeCount != valCount</code> |



## chap 6: Array

---

Consider the following instance variable `nums` and method `findLongest` with line numbers added for reference. Method `findLongest` is intended to find the longest consecutive block of the value `target` occurring in the array `nums`; however, `findLongest` does not work as intended.

For example, if the array `nums` contains the values `[7, 10, 10, 15, 15, 15, 15, 10, 10, 10, 15, 10, 10]`, the call `findLongest(10)` should return 3, the length of the longest consecutive block of 10s.

```
private int[] nums;

public int findLongest(int target)
{
 int lenCount = 0;
 int maxLen = 0;

Line 1: for (int val : nums)
Line 2: {
Line 3: if (val == target)
Line 4: {
Line 5: lenCount++;
Line 6: }
Line 7: else
Line 8: {
Line 9: if (lenCount > maxLen)
Line 10: {
Line 11: maxLen = lenCount;
Line 12: }
Line 13: }
Line 14: }
Line 15: if (lenCount > maxLen)
Line 16: {
Line 17: maxLen = lenCount;
Line 18: }
Line 19: return maxLen;
}
```

14. The method `findLongest` does not work as intended. Which of the following best describes the value returned by a call to `findLongest` ?
- (A) It is the length of the shortest consecutive block of the value `target` in `nums`.
  - (B) It is the length of the array `nums`.
  - (C) It is the number of occurrences of the value `target` in `nums`.
  - (D) It is the length of the first consecutive block of the value `target` in `nums`.
  - (E) It is the length of the last consecutive block of the value `target` in `nums`.
15. Which of the following changes should be made so that method `findLongest` will work as intended?

**chap 6: Array**

- (A) Insert the statement `lenCount = 0;`  
between lines 2 and 3.
  - (B) Insert the statement `lenCount = 0;`  
between lines 8 and 9.
  - (C) Insert the statement `lenCount = 0;`  
between lines 10 and 11.
  - (D) Insert the statement `lenCount = 0;`  
between lines 11 and 12.
  - (E) Insert the statement `lenCount = 0;`  
between lines 12 and 13.
- 

16. Consider the following instance variable and method.

```
private int[] arr;

/** Precondition: arr.length > 0
 * @return the largest value in array arr
 */
public int findMax()
{
 int maxVal = 0;

 for (int val : arr)
 {
 if (val > maxVal)
 {
 maxVal = val;
 }
 }

 return maxVal;
}
```

Method `findMax` is intended to return the largest value in the array `arr`. Which of the following best describes the conditions under which the method `findMax` will not work as intended?

- (A) The largest value in `arr` occurs only once and is in `arr[0]`.
- (B) The largest value in `arr` occurs only once and is in `arr[arr.length - 1]`.
- (C) The largest value in `arr` is negative.
- (D) The largest value in `arr` is zero.
- (E) The largest value in `arr` occurs more than once.

**chap 6: Array**

17. The following incomplete method is intended to return the largest integer in the array `numbers`.

```
// precondition: numbers.length > 0

public static int findMax(int[] numbers)
{
 int posOfMax = 0;

 for (int index = 1; index < numbers.length; index++)
 {
 if (/*condition*/)
 {
 /* statement */
 }
 }
 return numbers[posOfMax];
}
```

Which of the following can be used to replace `/* condition */` and `/* statement */` so that `findMax` will work as intended?

## chap 6: Array

(A)

| <u><i>/* condition */</i></u>      | <u><i>/* statement */</i></u> |
|------------------------------------|-------------------------------|
| numbers[index] > numbers[posOfMax] | posOfMax = numbers[index];    |

(B)

| <u><i>/* condition */</i></u>      | <u><i>/* statement */</i></u> |
|------------------------------------|-------------------------------|
| numbers[index] > numbers[posOfMax] | posOfMax = index;             |

(C)

| <u><i>/* condition */</i></u> | <u><i>/* statement */</i></u> |
|-------------------------------|-------------------------------|
| numbers[index] > posOfMax     | posOfMax = numbers[index];    |

(D)

| <u><i>/* condition */</i></u> | <u><i>/* statement */</i></u> |
|-------------------------------|-------------------------------|
| numbers[index] < posOfMax     | posOfMax = numbers[index];    |

(E)

| <u><i>/* condition */</i></u>      | <u><i>/* statement */</i></u> |
|------------------------------------|-------------------------------|
| numbers[index] < numbers[posOfMax] | posOfMax = index;             |

**chap 6: Array**

18. Consider the following incomplete method. Method `findNext` is intended to return the index of the first occurrence of the value `val` beyond the position `start` in array `arr`.

```
// returns index of first occurrence of val in arr

// after position start;

// returns arr.length if val is not found

public int findNext(int[] arr, int val, int start)

{

 int pos = start + 1;

 while (/* condition */)

 pos++;

 return pos;

}
```

For example, consider the following code segment.

```
int[] arr = {11, 22, 100, 33, 100, 11, 44, 100};

System.out.println(findNext(arr, 100, 2));
```

The execution of the code segment should result in the value 4 being printed.

Which of the following expressions could be used to replace `/* condition */` so that `findNext` will work as intended?

**chap 6: Array**

- (A) `(pos < arr.length) && (arr[pos] != val)`
- (B) `(arr[pos] != val) && (pos < arr.length)`
- (C) `(pos < arr.length) || (arr[pos] != val)`
- (D) `(arr[pos] == val) && (pos < arr.length)`
- (E) `(pos < arr.length) || (arr[pos] == val)`

19. Consider the code segment below, where `arr` is a one-dimensional array of integers.

```
int sum = 0;
for (int n : arr)
{
 sum = sum + 2 * n;
}
System.out.print(sum);
```

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

- (A) 

```
int sum = 0;
for (int k = 0; k < arr.length; k++)
{
 sum = sum + 2 * k;
}
System.out.print(sum);
```
- (B) 

```
int sum = 0;
for (int k = 0; k <= arr.length; k++)
{
 sum = sum + 2 * k;
}
System.out.print(sum);
```
- (C) 

```
int sum = 0;
for (int k = 1; k <= arr.length; k++)
{
 sum = sum + 2 * k;
}
System.out.print(sum);
```
- (D) 

```
int sum = 0;
for (int k = 0; k < arr.length; k++)
{
 sum = sum + 2 * arr[k];
}
System.out.print(sum);
```
- (E) 

```
int sum = arr[0];
for (int k = 1; k <= arr.length; k++)
{
 sum = sum + 2 * arr[k];
}
System.out.print(sum);
```

**chap 6: Array**

20. The code segment below is intended to print the length of the shortest string in the array `wordArray`. Assume that `wordArray` contains at least one element.

```
int shortest = /* missing value */;
for (String word : wordArray)
{
 if (word.length() < shortest)
 {
 shortest = word.length();
 }
}
System.out.println(shortest);
```

Which of the following should be used as the initial value assigned to `shortest` so that the code segment works as intended?

- (A) `Integer.MAX_VALUE`
  - (B) `Integer.MIN_VALUE`
  - (C) `0`
  - (D) `word.length()`
  - (E) `wordArray.length`
21. Assume that an array of integer values has been declared as follows and has been initialized.

```
int[] arr = new int[10];
```

Which of the following code segments correctly interchanges the value of `arr[0]` and `arr[5]` ?

- (A) `arr[0] = 5;`  
`arr[5] = 0;`
- (B) `arr[0] = arr[5];`  
`arr[5] = arr[0];`
- (C) `int k = arr[5];`  
`arr[0] = arr[5];`  
`arr[5] = k;`
- (D) `int k = arr[0];`  
`arr[0] = arr[5];`  
`arr[5] = k;`
- (E) `int k = arr[5];`  
`arr[5] = arr[0];`  
`arr[0] = arr[5];`

**chap 6: Array**

22. Consider the following method that is intended to return true if an array of integers is arranged in decreasing order and return false otherwise.

```
/** @param nums an array of integers

 * @return true if the values in the array appear in decreasing order
 * false otherwise

 */

public static boolean isDecreasing(int[] nums)

{

 /* missing code */

}
```

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

I. `for (int k = 0; k < nums.length; k++)`

`{`

`if (nums[k] <= nums[k + 1])`

`return false;`

`}`

`return true;`

II. `for (int k = 1; k < nums.length; k++)`

`{`



**chap 6: Array**

```
 if (nums[k - 1] <= nums[k])
```

```
 return false;
```

```
 }
```

```
 return true;
```

III. for (int k = 0; k < nums.length - 1; k++)

```
{
```

```
 if (nums[k] <= nums[k + 1])
```

```
 return false;
```

```
 else
```

```
 return true;
```

```
}
```

```
 return true;
```

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

**chap 6: Array**

23. 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

**chap 6: Array**

24. Consider the following class definition.

```
public class Book
{
 private int pages;
 public int getPages()
 {
 return pages;
 }
 // There may be instance variables, constructors, and methods not
 shown.
}
```

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

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

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

- (A) 

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

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

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

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

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

**chap 6: Array**

25. On Sunday night, a meteorologist records predicted daily high temperatures, in degrees Fahrenheit, for the next seven days. At the end of each day, the meteorologist records the actual daily high temperature, in degrees Fahrenheit. At the end of the seven-day period, the meteorologist would like to find the greatest absolute difference between a predicted temperature and a corresponding actual temperature.

Consider the following method, which is intended to return the greatest absolute difference between any pair of corresponding elements in the `int` arrays `pred` and `act`.

```
/** Precondition: pred and act have the same non-zero length. */
public static int diff(int[] pred, int[] act)
{
 int num = Integer.MIN_VALUE;
 for (int i = 0; i < pred.length; i++)
 {
 /* missing code */
 }
 return num;
}
```

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

- (A) 

```
if (pred[i] < act[i])
{
 num = act[i] - pred[i];
}
```
- (B) 

```
if (pred[i] > act[i])
{
 num = pred[i] - act[i];
}
```
- (C) 

```
if (pred[i] - act[i] > num)
{
 num = pred[i] - act[i];
}
```
- (D) 

```
if (Math.abs(pred[i] - act[i]) < num)
{
 num = Math.abs(pred[i] - act[i]);
}
```
- (E) 

```
if (Math.abs(pred[i] - act[i]) > num)
{
 num = Math.abs(pred[i] - act[i]);
}
```

**chap 6: Array**

26. Consider the following incomplete method, which is intended to return the longest string in the string array `words`. Assume that the array contains at least one element.

```
public static String longestWord(String[] words)
{
 /* missing declaration and initialization */
 for (int k = 1; k < words.length; k++)
 {
 if (words[k].length() > longest.length())
 {
 longest = words[k];
 }
 }
 return longest;
}
```

Which of the following can replace `/* missing declaration and initialization */` so that the method will work as intended?

- (A) `int longest = 0;`
- (B) `int longest = words[0].length();`
- (C) `String longest = "";`
- (D) `String longest = words[0];`
- (E) `String longest = words[1];`

27. Consider the following code segment.

```
int[] arr = {1, 2, 3, 4, 5, 6, 7};
for (int i = 1; i < arr.length; i += 2)
{
 arr[i] = arr[i - 1];
}
```

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

- (A) `{0, 1, 2, 3, 4, 5, 6}`
- (B) `{1, 1, 1, 1, 1, 1, 1}`
- (C) `{1, 1, 3, 3, 5, 5, 7}`
- (D) `{1, 2, 3, 4, 5, 6, 7}`
- (E) `{2, 2, 4, 4, 6, 6, 7}`

**chap 6: Array**

28. Consider the following instance variable and method.

```
private int[] numbers;

/** Precondition: numbers contains int values in no particular order.
 */
public int mystery(int num)
{
 for (int k = numbers.length - 1; k >= 0; k--)
 {
 if (numbers[k] < num)
 {
 return k;
 }
 }
 return -1;
}
```

Which of the following best describes the contents of numbers after the following statement has been executed?

`int m = mystery(n);`

- (A) All values in positions 0 through m are less than n.
- (B) All values in positions m+1 through numbers.length-1 are less than n.
- (C) All values in positions m+1 through numbers.length-1 are greater than or equal to n.
- (D) The smallest value is at position m.
- (E) The largest value that is smaller than n is at position m.

**chap 6: Array**

29. 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
30. Consider the following instance variable and method.

```
private int[] numbers;

public void mystery(int x)
{
 for (int k = 1; k < numbers.length; k = k + x)
 {
 numbers[k] = numbers[k - 1] + x;
 }
}
```

Assume that `numbers` has been initialized with the following values.

{17, 34, 21, 42, 15, 69, 48, 25, 39}

Which of the following represents the order of the values in `numbers` as a result of the call `mystery(3)`?

**chap 6: Array**

- (A) {17, 20, 21, 42, 45, 69, 48, 51, 39}
- (B) {17, 20, 23, 26, 29, 32, 35, 38, 41}
- (C) {17, 37, 21, 42, 18, 69, 48, 28, 39}
- (D) {20, 23, 21, 42, 45, 69, 51, 54, 39}
- (E) {20, 34, 21, 45, 15, 69, 51, 25, 39}

31. Consider the following method.

```
/** Precondition: arr.length > 0 */
public static int mystery(int[] arr)
{
 int index = 0;
 int count = 0;
 int m = -1;

 for (int outer = 0; outer < arr.length; outer++)
 {
 count = 0;
 for (int inner = outer + 1; inner < arr.length; inner++)
 {
 if (arr[outer] == arr[inner])
 {
 count++;
 }
 }

 if (count > m)
 {
 index = outer;
 m = count;
 }
 }

 return index;
}
```

Assume that `nums` has been declared and initialized as an array of integer values. Which of the following best describes the value returned by the call `mystery(nums)` ?

- (A) The maximum value that occurs in `nums`
- (B) An index of the maximum value that occurs in `nums`
- (C) The number of times that the maximum value occurs in `nums`
- (D) A value that occurs most often in `nums`
- (E) An index of a value that occurs most often in `nums`



**chap 6: Array**

32. The question refer to the following data field and method.

```
private int[] arr;

// precondition: arr.length > 0
public void mystery()

{

 int s1 = 0;

 int s2 = 0;

 for (int k = 0; k < arr.length; k++)

 {

 int num = arr[k];

 if ((num > 0) && (num % 2 == 0))

 s1 += num;

 else if (num < 0)

 s2 += num;

 }

 System.out.println(s1);

 System.out.println(s2);

}
```

Which of the following best describes the value of **s1** output by the method **mystery** ?

- (A) The sum of all positive values in **arr**
- (B) The sum of all positive even values in **arr**
- (C) The sum of all positive odd values in **arr**
- (D) The sum of all values greater than 2 in **arr**
- (E) The sum of all values less than 2 in **arr**

**chap 6: Array**

33. The question refer to the following data field and method.

```
private int[] arr;

// precondition: arr.length > 0
public void mystery()

{

 int s1 = 0;

 int s2 = 0;

 for (int k = 0; k < arr.length; k++)

 {

 int num = arr[k];

 if ((num > 0) && (num % 2 == 0))

 s1 += num;

 else if (num < 0)

 s2 += num;

 }

 System.out.println(s1);

 System.out.println(s2);

}
```

Which of the following best describes the value of **s2** output by the method **mystery** ?

- (A) The sum of all positive values in **arr**
- (B) The sum of all positive even values in **arr**
- (C) The sum of all negative values in **arr**
- (D) The sum of all negative even values in **arr**
- (E) The sum of all negative odd values in **arr**

**chap 6: Array**

34. Consider the following method.

```
public static int mystery(int[] arr)
{
 int x = 0;

 for (int k = 0; k < arr.length; k = k + 2)
 x = x + arr[k];

 return x;
}
```

Assume that the array `nums` has been declared and initialized as follows.

```
int [] nums = { 3, 6, 1, 0, 1, 4, 2};
```

What value will be returned as a result of the call `mystery(nums)` ?

- (A) 5
  - (B) 6
  - (C) 7
  - (D) 10
  - (E) 17
35. Consider the following code segment.

```
int[] arr = {7, 2, 5, 3, 0, 10};
for (int k = 0; k < arr.length - 1; k++)
{
 if (arr[k] > arr[k + 1])
 System.out.print(k + " " + arr[k] + " ");
}
```

What will be printed as a result of executing the code segment?

- (A) 0 2 2 3 3 0
- (B) 0 7 2 5 3 3
- (C) 0 7 2 5 5 10
- (D) 1 7 3 5 4 3
- (E) 7 2 5 3 3 0

**chap 6: Array**

36. Consider the following instance variable, `arr`, and incomplete method, `partialSum`. The method is intended to return an integer array `sum` such that for all `k`, `sum [ k ]` is equal to `arr[0] + arr[1] + ... + arr[k]`. For instance, if `arr` contains the values { 1, 4, 1, 3 }, the array `sum` will contain the values { 1, 5, 6, 9 }.

```
private int[] arr;
public int[] partialSum()
{
 int[] sum = new int[arr.length];
 for (int j = 0; j < sum.length; j++)
 {
 sum[j] = 0;
 }
 /* missing code */
 return sum;
}
```

The following two implementations of `/* missing code */` are proposed so that `partialSum` will work as intended.

**Implementation 1**

```
for (int j = 0; j < arr.length; j++)
{
 sum[j] = sum[j - 1] + arr[j];
}
```

**Implementation 2**

```
for (int j = 0; j < arr.length; j++)
{
 for (int k = 0; k <= j; k++)
 {
 sum[j] = sum[j] + arr[k];
 }
}
```

Which of the following statements is true?

- (A) Both implementations work as intended, but implementation 1 is faster than implementation 2.
- (B) Both implementations work as intended, but implementation 2 is faster than implementation 1.
- (C) Both implementations work as intended and are equally fast.
- (D) Implementation 1 does not work as intended, because it will cause an `ArrayIndexOutOfBoundsException`.
- (E) Implementation 2 does not work as intended, because it will cause an `ArrayIndexOutOfBoundsException`.

**chap 6: Array**

37. Consider the following method.

```
public int[] addNum(int[] array, int first, int second, int num)
{
 int[] newArray = new int[array.length];
 newArray[first] = array[first] + num;
 newArray[second] = array[second] + num;
 return newArray;
}
```

Which of the following code segments, appearing in the same class as the `addNum` method, will result in `array2` having the contents `{0, 0, 13, 0, 9, 0, 0}`?

- (A) `int[] array1 = {5, 2, 8, 6, 4, 3, 9};`  
`int[] array2 = addNum(array1, 2, 4, 5);`
  - (B) `int[] array1 = {-5, -5, 13, 0, 9, 0, 0};`  
`int[] array2 = addNum(array1, 2, 4, 5);`
  - (C) `int[] array1 = {5, 2, 8, 6, 4, 3, 9};`  
`int[] array2 = addNum(array1, 3, 5, 5);`
  - (D) `int[] array1 = {5, 8, 2, 4, 6, 3, 9};`  
`int[] array2 = addNum(array1, 2, 4, 5);`
  - (E) `int[] array1 = {0, -5, 8, 0, 9, 0, 0};`  
`int[] array2 = addNum(array1, 2, 4, 5);`
38. Consider the following method, which is intended to return the average (arithmetic mean) of the values in an integer array. Assume the array contains at least one element.

```
public static double findAvg(double[] values)
{
 double sum = 0.0;
 for (double val : values)
 {
 sum += val;
 }
 return sum / values.length;
}
```

Which of the following preconditions, if any, must be true about the array `values` so that the method works as intended?

- (A) The array `values` must be sorted in ascending order.
- (B) The array `values` must be sorted in descending order.
- (C) The array `values` must have only one mode.
- (D) The array `values` must not contain values whose sum is not 0.
- (E) No precondition is necessary; the method will always work as intended.

**chap 6: Array**

39. Consider the following code segment.

```
int[] arr = {4, 2, 9, 7, 3};

for (int k : arr)

{

 k = k + 10;

 System.out.print(k + " ");

}

for (int k : arr)

 System.out.print(k + " ");
```

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

- (A) 0 1 2 3 4 0 1 2 3 4
  - (B) 4 2 9 7 3 4 2 9 7 3
  - (C) 10 11 12 13 14 0 1 2 3 4
  - (D) 14 12 19 17 13 4 2 9 7 3
  - (E) 14 12 19 17 13 14 12 19 17 13
40. Consider the following method.

```
public static int getValue(int[] data, int j, int k)
{
 return data[j] + data[k];
}
```

Which of the following code segments, when appearing in another method in the same class as `getValue`, will print the value 70 ?

**chap 6: Array**

- (A) `int arr = {40, 30, 20, 10, 0};`  
`System.out.println(getValue(arr, 1, 2));`
- (B) `int[] arr = {40, 30, 20, 10, 0};`  
`System.out.println(getValue(arr, 1, 2));`
- (C) `int[] arr = {50, 40, 30, 20, 10};`  
`System.out.println(getValue(arr, 1, 2));`
- (D) `int arr = {40, 30, 20, 10, 0};`  
`System.out.println(getValue(arr, 2, 1));`
- (E) `int arr = {50, 40, 30, 20, 10};`  
`System.out.println(getValue(arr, 2, 1));`

41. Assume that the array `arr` has been defined and initialized as follows.

```
int[] arr = /* initial values for the array */ ;
```

Which of the following will correctly print all of the odd integers contained in `arr` but none of the even integers contained in `arr` ?

- (A) 

```
for (int x : arr)
 if (x % 2 != 0)
 System.out.println(x);
```
- (B) 

```
for (int k = 1; k < arr.length; k++)
 if (arr[k] % 2 != 0)
 System.out.println(arr[k]);
```
- (C) 

```
for (int x : arr)
 if (x % 2 != 0)
 System.out.println(arr[x]);
```
- (D) 

```
for (int k = 0; k < arr.length; k++)
 if (arr[k] % 2 != 0)
 System.out.println(k);
```
- (E) 

```
for (int x : arr)
 if (arr[x] % 2 != 0)
 System.out.println(arr[x]);
```

**chap 6: Array**

42. Consider the following method, which is intended to return an array of integers that contains the elements of the parameter `arr` arranged in reverse order. For example, if `arr` contains `{7, 2, 3, -5}`, then a new array containing `{-5, 3, 2, 7}` should be returned and the parameter `arr` should be left unchanged.

```
public static int[] reverse(int[] arr)
{
 int[] newArr = new int[arr.length];
 for (int k = 0; k < arr.length; k++)
 {
 /* missing statement */
 }
 return newArr;
}
```

Which of the following statements can be used to replace `/* missing statement */` so that the method works as intended?

- (A) `newArray[k] = arr[-k];`
  - (B) `newArray[k] = arr[k - arr.length];`
  - (C) `newArray[k] = arr[k - arr.length - 1];`
  - (D) `newArray[k] = arr[arr.length - k];`
  - (E) `newArray[k] = arr[arr.length - k - 1];`
43. Consider the following code segment.

```
int[] arr = {1, 2, 3, 4, 5, 6, 7};

for (int k = 3; k < arr.length - 1; k++)
 arr[k] = arr[k + 1];
```

Which of the following represents the contents of `arr` as a result of executing the code segment?

- (A) `{1, 2, 3, 4, 5, 6, 7}`
- (B) `{1, 2, 3, 5, 6, 7}`
- (C) `{1, 2, 3, 5, 6, 7, 7}`
- (D) `{1, 2, 3, 5, 6, 7, 8}`
- (E) `{2, 3, 4, 5, 6, 7, 7}`



## chap 6: Array

44. Consider a shuffle method that is intended to return a new array that contains all the elements from `nums`, but in a different order. Let  $n$  be the number of elements in `nums`. The shuffle method should alternate the elements from `nums[0] ... nums[n / 2 - 1]` with the elements from `nums[n / 2] ... nums[n - 1]`, as illustrated in the following examples.

Example 1

|        |    |    |    |    |    |    |    |    |
|--------|----|----|----|----|----|----|----|----|
|        | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
| nums   | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 |
|        | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  |
| result | 10 | 50 | 20 | 60 | 30 | 70 | 40 | 80 |

Example 2

|        |    |    |    |    |    |    |    |
|--------|----|----|----|----|----|----|----|
|        | 0  | 1  | 2  | 3  | 4  | 5  | 6  |
| nums   | 10 | 20 | 30 | 40 | 50 | 60 | 70 |
|        | 0  | 1  | 2  | 3  | 4  | 5  | 6  |
| result | 10 | 40 | 20 | 50 | 30 | 60 | 70 |

The following implementation of the shuffle method does not work as intended.

```
public static int[] shuffle(int[] nums)
{
 int n = nums.length;
 int[] result = new int[n];

 for (int j = 0; j < n / 2; j++)
 {
 result[j * 2] = nums[j];
 result[j * 2 + 1] = nums[j + n / 2];
 }

 return result;
}
```

Which of the following best describes the problem with the given implementation of the shuffle method?

**chap 6: Array**

- (A) Executing shuffle may cause an `ArrayIndexOutOfBoundsException`.
- (B) The first element of the returned array (`result[0]`) may not have the correct value.
- (C) The last element of the returned array (`result[result.length - 1]`) may not have the correct value.
- (D) One or more of `nums[0] ... nums[nums.length / 2 - 1]` may have been copied to the wrong position(s) in the returned array.
- (E) One or more of `nums[nums.length / 2] ... nums[nums.length - 1]` may have been copied to the wrong position(s) in the returned array.
45. Consider an integer array, `nums`, which has been declared and initialized with one or more integer values. Which of the following code segments updates `nums` so that each element contains the square of its original value?

**I.**

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

**II.**

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

**III.**

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

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

**chap 6: Array**

46. 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

**chap 6: Array**

47. Consider the following method that is intended to return the sum of the elements in the array `key`.

```
public static int sumArray(int[] key)
{
 int sum = 0;

 for (int i = 1; i <= key.length; i++)
 {
 /* missing code */
 }

 return sum;
}
```

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

- (A) `sum = key [ i ] ;`
  - (B) `sum += key [ i - 1 ] ;`
  - (C) `sum += key [ i ] ;`
  - (D) `sum += sum + key[i - 1] ;`
  - (E) `sum += sum + key [ i ] ;`
48. Consider the following code segment, which traverses two integer arrays of equal length. If any element of `arr1` is smaller than the corresponding (i.e., at the same index) element of `minArray`, the code segment should replace the element of `minArray` with the corresponding element of `arr1`. After the code segment executes, `minArray` should hold the smaller of the two elements originally found at the same indices in `arr1` and `minArray` and `arr1` should remain unchanged.

```
for (int c = 0; c < arr1.length; c++)
{
 if (arr1[c] < minArray[c])
 {
 arr1[c] = minArray[c];
 }
 else
 {
 minArray[c] = arr1[c];
 }
}
```

Which of the following changes will ensure that the code segment always works as intended?

**chap 6: Array**

- (A) Changing the Boolean expression in line 1 to `c <= arr1.length`
- (B) Changing the relational operator in line 3 to `>`
- (C) Removing lines 5–8
- (D) Swapping the positions of line 5 and line 9
- (E) Removing lines 7–10

49. Consider the following code segment.

```
boolean[] oldVals = {true, false, true, true};
boolean[] newVals = new boolean[4];
for (int j = oldVals.length - 1; j >= 0; j--)
{
 newVals[j] = !(oldVals[j]);
}
```

What, if anything, will be the contents of `newVals` as a result of executing the code segment?

- (A) `{true, true, false, true}`
- (B) `{true, false, true, true}`
- (C) `{false, true, false, false}`
- (D) `{false, false, true, false}`
- (E) The array `newVals` will not contain any values because the code segment does not compile.

**chap 6: Array**

50. Consider the following code segment.

```
int[] arr = {1, 2, 4, 0, 3};
for (int i : arr)
{
 System.out.print(i);
}
```

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

**I.**

```
int[] arr = {1, 2, 4, 0, 3};
for (int i : arr)
{
 System.out.print(arr[i]);
}
```

**II.**

```
int[] arr = {1, 2, 4, 0, 3};
for (int i = 0; i < arr.length; i++)
{
 System.out.print(i);
}
```

**III.**

```
int[] arr = {1, 2, 4, 0, 3};
for (int i = 0; i < arr.length; i++)
{
 System.out.print(arr[i]);
}
```

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

**chap 6: Array**

51. The `twoInARow` method below is intended to return `true` if any two consecutive elements of the parameter `arr` are equal in value and return `false` otherwise.

```
public boolean twoInARow(int[] arr)
{
 /* missing loop header */
 {
 if (arr[k] == arr[k + 1])
 {
 return true;
 }
 }
 return false;
}
```

Which of the following can be used to replace `/* missing loop header */` so that the method will work as intended?

- (A) `for (int k = 0; k < arr.length - 1; k++)`
  - (B) `for (int k = 0; k < arr.length; k++)`
  - (C) `for (int k = 1; k < arr.length; k++)`
  - (D) `for (int k = arr.length - 1; k >= 0; k--)`
  - (E) `for (int k = arr.length - 1; k > 0; k--)`
52. Consider the following code segment.

```
int[] arr = {3, 1, 0, 4, 2};
for(int j = 0; j < arr.length; j++)
{
 System.out.print(arr[j] + j + " ");
}
```

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

- (A) 3 1 0 4 2
- (B) 3 2 2 7 6
- (C) 6 2 0 8 4
- (D) 7 2 3 6 2
- (E) Nothing is printed, because an `ArrayIndexOutOfBoundsException` is thrown.