

1. Consider the following data field and method.

```
private ArrayList<Integer> nums;

// precondition: nums.size() > 0;
//      nums contains Integer objects
public void numQuest() {
    int k = 0;
    Integer zero = new Integer(0);

    while (k < nums.size()) {
        if (nums.get(k).equals(zero))
            nums.remove(k);

        k++;
    }
}
```

Assume that `ArrayList<Integer> nums` initially contains the following `Integer` values.

[0, 0, 4, 2, 5, 0, 3, 0]

What will `ArrayList<Integer> nums` contain as a result of executing `numQuest`?

- a) [0, 0, 4, 2, 5, 0, 3, 0]
- b) [4, 2, 5, 3]
- c) [0, 0, 0, 0, 4, 2, 5, 3]
- d) [3, 5, 2, 4, 0, 0, 0, 0]
- e) [0, 4, 2, 5, 3]

2. Consider the following code segment, applied to `list`, which is declared as an `ArrayList<Integer>`.

```
int len = list.size();
for (int i=0; i<len; i++) {
    list.add(i+1, new Integer(i));
    list.set(i, new Integer(i+2));
}
```

If `list` is initially 6 1 8, what will it be following execution of the code segment?

- a) 2 3 4 2 1 8
- b) 2 3 4 6 2 2 0 1 8
- c) 2 3 4 0 1 2
- d) 2 3 4 6 1 8
- e) 2 3 3 2

The next two questions refer to the following information.

Consider the following data field 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(nums, 10)` should return 3, the length of the longest consecutive block of 10's.

```
private int[] nums;
```

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

```
Line 1:     for (int k = 0; k < nums.length; k++) {  
Line 2:         if (nums[k] == target) {  
Line 3:             lenCount++;  
Line 4:         }  
Line 5:         else {  
Line 6:             if (lenCount > maxLen) {  
Line 7:                 maxLen = lenCount;  
Line 8:             }  
Line 9:         }  
Line 10:     }  
Line 11:     if (lenCount > maxLen) {  
Line 12:         maxLen = lenCount;  
Line 13:     }  
Line 14:     return maxLen;  
}
```

3. 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`.

4. Which of the following changes should be made so that method `findLongest` will work as intended?
- a) Insert the statement `lenCount = 0;` between lines 1 and 2.
 - b) Insert the statement `lenCount = 0;` between lines 5 and 6.
 - c) Insert the statement `lenCount = 0;` between lines 6 and 7.
 - d) Insert the statement `lenCount = 0;` between lines 7 and 8.
 - e) Insert the statement `lenCount = 0;` between lines 8 and 9.

5. Consider the following data field and method.

```
private int[] myStuff;  
  
// precondition: myStuff contains integers in no particular order  
public int mystery(int num) {  
    for (int k = myStuff.length-1; k >= 0; k--) {  
        if (myStuff[k] > num) {  
            return k;  
        }  
    }  
    return -1;  
}
```

Which of the following best describes the contents of `myStuff` 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 `myStuff.length-1` are less than `n`.
- c) All values in positions `m+1` through `myStuff.length-1` are less 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`.

6. Consider the static method `selectSort` shown below. Method `selectSort` is intended to sort an array into increasing order; however, it does not always work as intended.

```
// precondition: numbers.length > 0
// postcondition: numbers is orted in increasing order
public static void selectSort(int[] numbers) {
    int temp;
Line 1:    for (int j = 0; j < numbers.length - 1; j++) {
Line 2:        int pos = 0;
Line 3:        for (int k = j + 1; k < numbers.length; k++) {
Line 4:            if (numbers[k] < numbers[pos]) {
Line 5:                pos = k;
            }
        }
        temp = numbers[j];
        numbers[j] = numbers[pos];
        numbers[pos] = temp;
    }
}
```

Which of the following changes should be made so that `selectSort` will work as intended?

- a) Line 1 should be changed to

```
for (int j = 0; j < numbers.length - 2; j++) {
```

- b) Line 2 should be changed to

```
int pos = j;
```

- c) Line 3 should be changed to

```
for (int k = 0; k < numbers.length; k++) {
```

- d) Line 4 should be changed to

```
if (numbers[k] > numbers[pos]) {
```

- e) Line 5 should be changed to

```
k = pos;
```

7. The following code fragment is intended to find the smallest value in `arr[0]...arr[n-1]`.

```
//Precondition: arr[0]...arr[n-1] initialized with integers.
//               arr is an array, arr.length = n
//Postcondition: min = smallest value in arr[0]...arr[n-1]
int min = arr[0];
int i = 1;
while (i < n) {
    i++;
    if (arr[i] < min)
        min = arr[i];
}
```

This code is incorrect. For the segment to work as intended, which of the following modifications could be made?

I Change the line

```
int i = 1;
```

to

```
int i = 0;
```

II Change the body of the `while` loop to

```
if (arr[i] < min)
```

```
    min = arr[i];
```

```
i++
```

III Change the test for the `while` loop as follows:

```
while (i <= n)
```

- a) I only
- b) II only
- c) III only
- d) I and II only
- e) I, II, and III

8. Refer to the following class, containing the `mystery` method.

```
public class SomeClass {
    private int[] arr;

    //Constructor. Initializes arr to contain nonnegative
    //integers K such that 0 <= k <= 9
    public SomeClass() {
        /* Implementation not shown */
    }

    public int mystery() {
        int value = arr[0];
        for (int i=1; i < arr.length; i++) {
            value = value*10 + arr[i];
        }
        return value;
    }
}
```

Which best describes what the `mystery` method does?

- a) It sums the elements of `arr`.
- b) It sums the products $10 \cdot \text{arr}[0] + 10 \cdot \text{arr}[1] + \dots + 10 \cdot \text{arr}[\text{arr.length}-1]$.
- c) It builds an integer of the form $d_1 d_2 d_3 \dots d_n$, where $d_1 = \text{arr}[0]$, $d_2 = \text{arr}[1]$, \dots , $d_n = \text{arr}[\text{arr.length}-1]$.
- d) It builds an integer of the form $d_1 d_2 d_3 \dots d_n$, where $d_1 = \text{arr}[\text{arr.length}-1]$, $d_2 = \text{arr}[\text{arr.length}-2]$, \dots , $d_n = \text{arr}[0]$.
- e) It converts the elements of `arr` to base 10.

9. Consider the method `doTask` defined below.

```
public static int doTask(int[] a) {
    int index = 0;
    int soFar = 1;
    int count = 1;
    for (int k = 1; k < a.length; k++) {
        if (a[k] == a[k - 1]) {
            count++;
            if (count > soFar) {
                soFar = count;
                index = k;
            }
        }
        else {
            count = 1;
        }
    }
    return a[index];
}
```

When the following code segment is executed,

```
int[] arr = {1,2,3,3,3,3,4,2,2,2,2,2,2,2,5,6,6,6,6,6,6,6,4,7,8};
System.out.println(doTask(arr));
```

What is printed to the screen?

- a) 1
- b) 2
- c) 6
- d) 7
- e) 24

10. Refer to the following declarations:

```
String[] colors = {"red", "green", "black"};  
ArrayList<String> colorList = new ArrayList<String>();
```

Which of the following correctly adds the elements of the `colors` array to `colorList`? The final ordering of colors in `colorList` should be the same as in the `colors` array.

- I.

```
for (int i=0; i<colors.length; i++)  
    colorList.add(i, colors.get(i));
```
- II.

```
for (int i=0; i<colors.length; i++)  
    colorList.add(colors[i]);
```
- III.

```
for (int i=colors.length-1; i>=0; i--)  
    colorList.add(i, colors[i]);
```

- a) I only
- b) II only
- c) III only
- d) II and III only
- e) I, II, and III

11. Suppose an `ArrayList<Integer>` called `list` is initialized with `Integer` values. Which of the following will *not* cause an `IndexOutOfBoundsException` runtime error?

- a)

```
for (int i=0; i<=list.size(); i++)  
    list.set(i, 0);
```
- b)

```
list.add(list.size(), 0);
```
- c)

```
int i = list.get(list.size());
```
- d)

```
int i = list.remove(list.size());
```
- e)

```
list.add(-1, 0);
```


12. The following program segment is intended to find the index of the first negative integer in `arr[0]...arr[n-1]`, where `arr` is an array of `n` integers.

```
int i = 0;
while (arr[i] >= 0) {
    i++;
}
location = i;
```

This segment will work in intended

- a) Always.
- b) Never.
- c) Whenever `arr` contains at least one negative integer.
- d) Whenever `arr` contains at least one nonnegative integer.
- e) Whenever `arr` contains no negative integers.

13. Below is the code for the Binary Search algorithm.

```
// precondition: list is an instantiated array, and sorted in ascending order
// postcondition: the index of the key is returned. If not found -1 is returned
public static int binarySearch(int[] numbers, int key) {
    int low = 0;
    int high = numbers.length-1;
    int middle = (low + high) / 2;

    while (numbers[middle] != key && low <= high) {
        if (key < numbers[middle])
            high = middle - 1;
        else
            low = middle + 1;
        middle = (low + high) / 2;
    }

    if (numbers[middle] == key)
        return middle;
    else
        return -1;
}
```

Given this array, how many iterations through the while loop does it take to determine that 18 is not in the list?

| | | | | | | | | | | | | | | | | | | | | |
|--------|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Value: | 3 | 13 | 17 | 20 | 35 | 43 | 51 | 54 | 55 | 55 | 61 | 63 | 68 | 68 | 72 | 72 | 80 | 81 | 85 | 97 |
| Index: | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |