CSE 143X: Accelerated Computer Programming I/II

Code Analysis Exercises Solutions

```
1    public static void mystery1(int[] a) {
2       for (int i = 0; i < a.length - 1; i++) {
3          if (a[i] > a[i + 1]) {
4             a[i + 1] = a[i + 1] * 2;
5          }
6       }
7    }
```

What are the results of the following calls to `mystery1`?

- `mystery1({1, 7, 5, 6, 4, 14, 11});`

Solution:

`[1, 7, 10, 12, 8, 14, 22]`

- `mystery1({1, 2, 3, 4, 5});`

Solution:

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

```
1  public static String mystery2(String s, int i, boolean[] arr) {
2     String result = i + s + i;
3     result += s.charAt(2);
4     result += s.substring(2, 4);
5
6     arr[0] = i == 0;
7     arr[3] = i == 3;
8
9     return result;
10 }
11 public static caller() {
12    boolean[] arr = {true, true, true, true};
13    String s = "hello";
14    System.out.println(mystery2(s, 3, arr));
15    System.out.println(Arrays.toString(arr) + "|" + s);
16    System.out.println(mystery2(s, 0, arr));
17    System.out.println(Arrays.toString(arr) + "|" + s);
18 }
```

What is printed out in a call to `caller()`?

Solution:

```
>> 3hello3lll
>> [false, true, true, true]|hello
>> 0hello0lll
>> [true, true, true, false]|hello
```

```java
public static int uhoh2(int[] arr) {
   int left = 0;
   int right = 0;
   int i = 0;
   while (arr[i] < arr[i + 1]) {
      // Point A
      if (Math.abs(arr[i] - arr[i + 1]) == 1) {
         // Point B
         if (arr[i] >= arr[i + 1]) {
            right++;
            // Point C
         }
         else {
            left++;
            // Point D
         }

      }
      else {
         left = 0;
         right = 0;
         // Point E
      }
      i++;
   }
   return left + right;
}
```

At each "point", how can we relate:

- `arr[i]` and `arr[i + 1]`?
- `left` and 0?
- `right` and 0?
- `left` and `right`?

At the end of the program, what is `left + right`?

Solution:

Some key points:

- The while loop is really a for loop; note that it will fall off the end of the array. This would throw an `ArrayOutOfBoundsException`.
- At Point A, we know `arr[i] < arr[i + 1]`
- At Point B, we know `arr[i + 1] = arr[i] + 1`
- Point C is unreachable. We call this "dead code".
- At Point D, we know what we knew at Point B as well as that this is the only condition that can increase `left`.
- Since we must have *increasing* indexes right next to each other that are one apart in value, and this is the only place we increase the quantity, the method is counting these.
- If it ever finds two adjacent indexes without this property, it resets to zero.

```
1  public static int uhoh3(Scanner console) {
2     int prev = 0;
3     int count = 0;
4     int next = console.nextInt();
5     // Point A
6     while (next != 0) {
7        // Point B
8        if (next == prev) {
9           // Point C
10          count++;
11       }
12       prev = next;
13       next = console.nextInt();
14       // Point D
15    }
16    // Point E
17    return count;
18 }
```

	next == 0	prev == 0	next == prev
Point A	**S**	**A**	**S**
Point B	**N**	**N**	**S**
Point C	**N**	**N**	**A**
Point D	**S**	**S**	**S**
Point E	**A**	**S**	**S**