1. Code Trace. 2 points each. On array output differences in []'s or ,'s okay. For example {} instead of []

-2 for first occurrence of "'s that do not appear in output.
For example " `gymnastics nast`" is -2, ignore quotes on following answers.

A. gymnastics nast
B. No
C. 4*D -1
D. [5, 2, -1, 13, 4]
E. [false, true, false, true, false] (note, t/f or 0/1 is wrong)
F. [2, 4, 1, -9] [2, 4, 1, -9]
G. Syntax error or Compiler Error (just error -2)
H. [6, 5, 5]
I. [5, 6][3, 8, 5]
J. [2, 2, 6, 10, 7]
K. false true (note, t/f or 0/1 is wrong)
L. 3DES2012 DES2012
M. -1, 0, 1, 2, 3
N. Runtime error or Exception (just error -2)
O. Runtime error or Exception (just error -2)
P. [5, -3, 8, -2, 12]

2. Scanners - 15 points

```
public static double averageOfNonPositiveInts(Scanner sc) {
    int total = 0;
    int count = 0;
    while (sc.hasNext() ) {
        if (sc.hasNextInt()) {
            int temp = sc.nextInt();
            if(temp <= 0) {
                total += temp;
                count++;
            }
        } else {
            sc.next();
        }
    }
    double result = 1.0;
    if (count > 0) {
        result = 1.0 * total / count;
    }
    return result;
}
```

method header: 1 point
while loop: 2 points
check if next token is int: 2 points
read in next int and check non positive: 3 points
correctly skip non int tokens: 3 points

result calculated correctly: 3 points
return correct value: 1 point

common deductions:
new Scanner, -2
reading in too many ints / skipping over valid ints -3
not handling 1.0 case for no non positive ints, -1

Common problems:
not skipping non ints
multiple calls to nextInt thinking it always returns the same value
stopping at the first non int

3. Program Logic, 1 point each

The give code sorts the elements in the array into descending order.

	c != 0	i == j	list[i] < list[j]	j == 0
POINT A	**N**	**A**	**N**	**A**
POINT B	**S**	**S**	**S**	**N**
POINT C	**A**	**N**	**A**	**S**
POINT D	**S**	**S**	**S**	**S**

4. Strings - 15 points

Suggested solution:

```
public static String getChoppedString(String str, int start, int skip) {
    String result = "";
    for(int i = start; i < str.length(); i += skip) {
        result += str.charAt(i);
    }
    return result;
}
```

create empty result: 2 points
start at correct spot: 3 points

boolean test to keep going correct: 3 points
calculate next position correctly: 3 points
concat correctly: 3 points
return correct result: 1point

Common problem:
Mixing up / confusing number of chars to get and index in loop.

5. Arrays 1 - 12 points

Suggested Solution:

```
public static int sumOfGaps(int[] data) {
    int total = 0;
    for(int i = 1; i < data.length; i++) {
        total += data[i] - data[i - 1];
    }
    return total;
}
```

total variable: 1 point
loop through correct portion of array: 5 points (-3 if off by one error leading to ArrayIndexOutOfBoundsException)
calculate gap between consecutive elements correctly and add to total: 5 points
-2 if incorrectly use subscript
return result: 1 point

Use of non allowed classes or methods: -4 per occurrence

6. Arrays and Strings - 15 points

```
public static int numThatStartOrEndWithChar(String[] vals, char c) {
    int result = 0;
    for(int i = 0; i < vals.length; i++) {
        String str = vals[i];
        if(str != null && str.length() > 0) {
            char first = str.charAt(0);
            char last = str.charAt(str.length() - 1);
            if(first == c || last == c) {
                result++;
            }
        }
    }
    return result;
}
```

result variable: 1 point
loop through all elements correctly: 2 points

handle nulls correctly without error: 2 points
handle 0 length Strings correctly with no errors: 2 points
check first char of String correctly: 2 points
check last char of String correctly: 3 points
correct logic to increment counter: 2 points
return correct answer: 1 point

Common problems:

using .equals on primitive char

7. Arrays .

```
public static int[] removeTarget(int[] data, int tgt) {
    // count the number of elements equal to the target
    int resultSize = 0;
    for (int i = 0; i < data.length; i++) {
       if (data[i] != tgt) {
            resultSize++;
        }
    }

    int[] result = new int[resultSize];
    if (resultSize > 0) {
        // put the elements equal to the target in the result
        int indexResult = 0;
        for (int i = 0; i < data.length; i++) {
            if (data[i] != tgt) {
                result[indexResult] = data[i];
                indexResult++;
            }
        }
    }
    return result;
}
```

Correctly determine number of elements non equal to target value: 6 point
create new array of correct size: 1 point
place values in new array: 2 points
place values at correct indices in new array, keeping track of indices in original and result correctly: 5 points
return correct result: 1 point

Common problems:
altering original array
not tracking indices correctly
returning original array instead of copy. (okay if empty, but not other cases)