

Problems | Name: _____

0.1 Design Before Implementation

Reference example

Input: nonempty list of scores

Output: largest score

Rule: do not modify the input

Cases: one item, duplicates, negative values

0.1.1 - Explain

A maximum finder starts best at 0. Give a failing input and explain a correct initialization.

0.1.2 - Write code

Implement and demonstrate a maximum scan for the nonempty array `{-8, -3, -10}` without changing it.

Problems | Name: _____

0.2 Abstraction and Encapsulation

Reference example

```
class Meter {  
    private int value;  
    void add(int amount) { if (amount > 0) value += amount; }  
    int read() { return value; }  
}  
Meter m = new Meter();  
m.add(4); m.add(-2);  
System.out.println(m.read());
```

0.2.1 - Explain

Why can returning an internal mutable list from an accessor weaken encapsulation?

0.2.2 - Write code

Create a Counter with a private count and an increment method, then show two increments.

Problems | Name: _____

0.3 Testing and Debugging Workflow

Reference example

```
int[] values = {-4, -1, -9};
int best = values[0];
for (int value : values) best = Math.max(best, value);
System.out.println(best);
```

0.3.1 - Trace

Determine the exact output and explain the state changes.

```
int[] a = {1, 2, 3};
int sum = 0;
for (int i = 0; i < a.length - 1; i++) sum += a[i];
System.out.println(sum);
```

0.3.2 - Explain

Give boundary tests for a method counting positive array elements.

Problems | Name: _____

0.4 Technical Communication

Reference example

Claim: a full scan of n elements uses $O(n)$ time

Evidence: exactly n loop iterations

Space: $O(1)$ auxiliary storage

Precondition: input array is available

0.4.1 - Explain

Rewrite "this algorithm is fast" as a precise claim for a loop that visits n elements once.

0.4.2 - Explain

Why should a bug report include input, expected output, actual output, and reproduction steps?
