



[Course](#) [Progress](#)

[Course](#) > [Readings/Videos](#) > [Reading 5: Designing Specifications](#) > Questions

[< Previous](#)





















[Next >](#)

Questions

 [Bookmark this page](#)

show me a sign

1/1 point (graded)

Which of the following are signs of an excellent specification (check all that apply):

- ☒ 1. the specification is declarative
- ☐ 2. the specification is operational
- ☐ 3. the specification is as super-strong as possible
- ☐ 4. the specification is as super-weak as possible
- ☐ 5. the implementation is allowed to ignore invalid arguments
- ☒ 6. the implementation is allowed to use different algorithms depending on the arguments
- ☐ 7. the specification utilizes the reader's knowledge of the implementation



Explanation

We prefer declarative specs (option 1) to operational ones (2).

We want a spec that is neither too strong (3) nor too weak (4) to balance the constraints of implementor and client.

We would rather have clear specs and implementations that fail fast than allow the implementation to quietly fail (5).

And we do not want the client to have to read the implementation at all (7).

[Submit](#)

 [Show Answer](#)

 Answers are displayed within the problem

that's an odd way of looking at it

1/1 point (graded)

```
public static int secondToLastIndexOf(int[] arr, int val)
  requires: val appears in arr an odd number of times
  effects: returns the 2nd-largest i such that arr[i] == val
```

Which of the following are reasonable criticisms of this spec? Check all that apply.

☒ The spec is not well-defined, we cannot implement it

☒ The spec is not coherent

☐ The spec is not deterministic

☐ The spec is not operational



Submit

Show Answer

Correct (1/1 point)

behavioral oddities

4/4 points (graded)

Consider the following test cases for `secondToLastIndexOf`:

`[1, 3, 4], 3` returns `1`

☐ valid test case

☐ could be valid with a weaker precondition, same postcondition

☐ could be valid with a weaker precondition, stronger postcondition

☒ could be valid with same precondition, weaker postcondition



Explanation

The current postcondition doesn't admit any output value with only one occurrence of `val`.

`[1, 3, 3, 4], 3` returns `1`

☐ valid test case

☒ could be valid with a weaker precondition, same postcondition

☐ could be valid with a weaker precondition, stronger postcondition

☐ could be valid with same precondition, stronger postcondition



Explanation

We would need to weaken the precondition to allow even occurrences of `val`.

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

☒ valid test case

☐ could be valid with a weaker precondition, same postcondition

☐ could be valid with a weaker precondition, stronger postcondition

☐ could be valid with same precondition, stronger postcondition



Explanation

Satisfies the precondition, and the postcondition is deterministic.

[3, 3, 3, 3], 3 throws an exception

☐ valid test case

☐ could be valid with a weaker precondition, same postcondition

☒ could be valid with a weaker precondition, stronger postcondition

☐ could be valid with same precondition, stronger postcondition



Explanation

This is not (currently) a valid test because it violates the precondition; maybe the method throws an exception in such cases, but that behavior is beyond the spec. So at least we need to weaken the precondition to allow this input. Having done that, we also need to change the postcondition to specify that an exception is thrown when `val` appears an even number of times, because the current postcondition doesn't mention any exceptions at all.

Submit

Show Answer

Answers are displayed within the problem

odd doc

1/1 point (graded)

Here's the spec again:

```
public static int secondToLastIndexOf(int[] arr, int val)
    requires: val appears in arr an odd number of times
    effects: returns the 2nd-largest i such that arr[i] == val
```

Choose lines below to complete one possible Javadoc version of this terrible spec:

☐ /*

☒ /**

☒ * Finds the second-to-last occurrence of a value in an array.

☐ * Find j, the largest index such that arr[j] == val.

☐ * Then find i, the largest index such that i < j and arr[i] == val.

☒ * @param arr array to search

☐ * @param arr fixed-size array of integers to search

☐ * @param val value to search for

☒ * @param val value to search for, requires val appears in arr an odd number of times

☐ * @return index i

☒ * @return second-largest index i such that arr[i] == val

☒ */



Explanation

The business about computing `j` and then `i` in terms of `j` is operational so we don't need that.

We don't need to explain Java semantics (e.g. `arr` is a fixed-size array of integers), but we definitely do need to explain the precondition.

It's good to include a summary of the function as the first line of the Javadoc, but we should still explain the postcondition on the return value clearly and succinctly.

And let's just be clear: regardless of how well we write it up, *this is a terrible specification*.

Submit

Show Answer

Answers are displayed within the problem

< Previous

Next >

Some Rights Reserved

[Open Learning Library](#)

[About](#)

[Accessibility](#)

[All Courses](#)

[Why Support MIT Open Learning?](#)

[Help](#)

[Connect](#)

[Contact](#)

[Twitter](#)

[Facebook](#)

[Privacy Policy](#) [Terms of Service](#)

© Massachusetts Institute of Technology, 2024