Questions

Bookmark this page

assertions

2/2 points (graded)

Consider this (incomplete) function:

```
/**
 * Solves quadratic equation ax^2 + bx + c = 0.
 *
 * @param a quadratic coefficient, requires a != 0
 * @param b linear coefficient
 * @param c constant term
 * @return a list of the real roots of the equation
 */
public static List<Double> quadraticRoots(final int a, final int b, final int c) {
    List<Double> roots = new ArrayList<Double>();
    // A
    ... // compute roots
    // B
    return roots;
}
```

What statements would be reasonable to write at position A? Check all that apply.

- [x] assert a != 0;
- [ ] assert b != 0;
- [ ] assert c != 0;
- [ ] assert roots.size() >= 0;
- [ ] assert roots.size() <= 2;
- [ ] roots.forEach((x) -> { assert Math.abs(a*x*x + b*x + c) < 0.0001; });

✔

What statements would be reasonable to write at position B? Check all that apply.

- [ ] assert a != 0;

- [ ] assert b != 0;
- [ ] assert c != 0;
- [ ] assert roots.size() >= 0;
- [x] assert roots.size() <= 2;
- [x] roots.forEach((x) -> { assert Math.abs(a*x*x + b*x + c) < 0.0001; });

✔

Note: roots.forEach() applies a lambda expression to each element of the roots list. Its rough equivalent in Python is:

```
map(lambda x: assert abs(a*x*x + b*x + c) < 0.0001, roots)
```

or:

```
for x in roots:
   assert abs(a*x*x + b*x + c) < 0.0001
```

Explanation

It's good to assert a != 0 at the start of the function, but not necessary to assert at the end of the function, since final prevents it from being changed.
It isn't correct to assert b!=0 or c!=0, since 0 is a legal value for those parameters.
It isn't reasonable to assert `roots.size()>=0` because lists are guaranteed to have nonnegative length. This assertion would only fail if the List `size()` method were broken. Don't assert guaranteed behavior of the language or library.
A quadratic equation can have at most 2 roots, so it's reasonable to assert `roots.size()<=2` .
It isn't reasonable to assert the content of the roots list until it has actually been computed -- i.e. not at the start of the function, but at the end of the function.

Submit

Show Answer

Answers are displayed within the problem

Previous Next