Quality: for(;;) {test; specify; code}

Illustrated with the simple problem of

testing the equivalence of circular lists

September 2016, [Pierre-Edouard Portier]

http://liris.cnrs.fr/pierre-edouard.portier/teaching_2016_2017/

$$A(i: 0 \leq i < N)$$
$$B(i: 0 \leq i < N)$$
$$A.i = A.(i + N)$$

```java
@Test
public void testSetGetModulo() {
    CL a = new CL(3);
    a.set(0, 7); a.set(4, 8); a.set(2, 9);
    assertEquals(8,a.get(1));
}
```

```java
@Invariant("s>=0")
public class CL {
    /**
     * s: size of the circular list (CL)
     */
    public final int s;
    /**
     * a: array used to model a CL
     */
    private int a[];

    CL(int _size){
        s = _size;
        a = new int[s];
    }
```

```java
/**
 * set the value of an element of the CL
 *
 * @param _i index in CL understood modulo the size of the CL
 * @param _x value of the new element
 */
@Requires("_i >= 0")
public void set(int _i, int _x){
    a[(_i%s)] = _x;
}
```

```java
/**
 * get a value from the CL
 *
 * @param _i index in CL understood modulo the size of the CL
 * @return the value at index _i modulo the size of the CL
 */
@Requires("_i >= 0")
public int get(int _i){
    return a[(_i%s)];
}
```

Set of rotations of A:

$$RA.i \;=\; A(k{:}\, i \le k < i + N)$$
$$RA.i \;= RA.(i + N)$$
$$\#RA \;\le N$$

```java
@Test
public void testRot() {
    CL a = new CL(3);
    a.set(0, 7); a.set(1, 8); a.set(2, 9);
    int[] expected = {8, 9, 7};
    assertArrayEquals(expected, a.rot(1));
}
```

```java
/**
 * rotate a CL by an offset of _i
 *
 * @param _i
 * @return an array that represents the rotation of the CL
 */
@Requires("_i >= 0")
public int[] rot(int _i){
    int[] res = new int[s];
    for(int j=_i, k=0 ; j < (_i+s) ; j++, k++){
        res[k] = get(j);
    }
    return res;
}
```

Postcondition:

$$R: \quad res \equiv \ (\exists i, j :: RA.i = RB.j)$$

```java
@Test
public void testRot_eq() {
    CL a = new CL(3);
    a.set(0,7); a.set(1,9); a.set(2,8);
    CL b = new CL(3);
    b.set(0,9); b.set(1,8); b.set(2,7);
    assertTrue(CL.rot_eq(a, 0, b, 2));
}
```

```java
/**
 * @param _a CL
 * @param _i offset used to rotate _a
 * @param _b CL
 * @param _j offset used to rotate _b
 * @return Is the _ith-rotation of CL _a equals to the _jth-rotation of CL _b?
 */
@Requires({"_i >= 0", "_j >= 0"})
public static boolean rot_eq(CL _a, int _i, CL _b, int _j){
    return Arrays.equals(_a.rot(_i), _b.rot(_j));
}
```

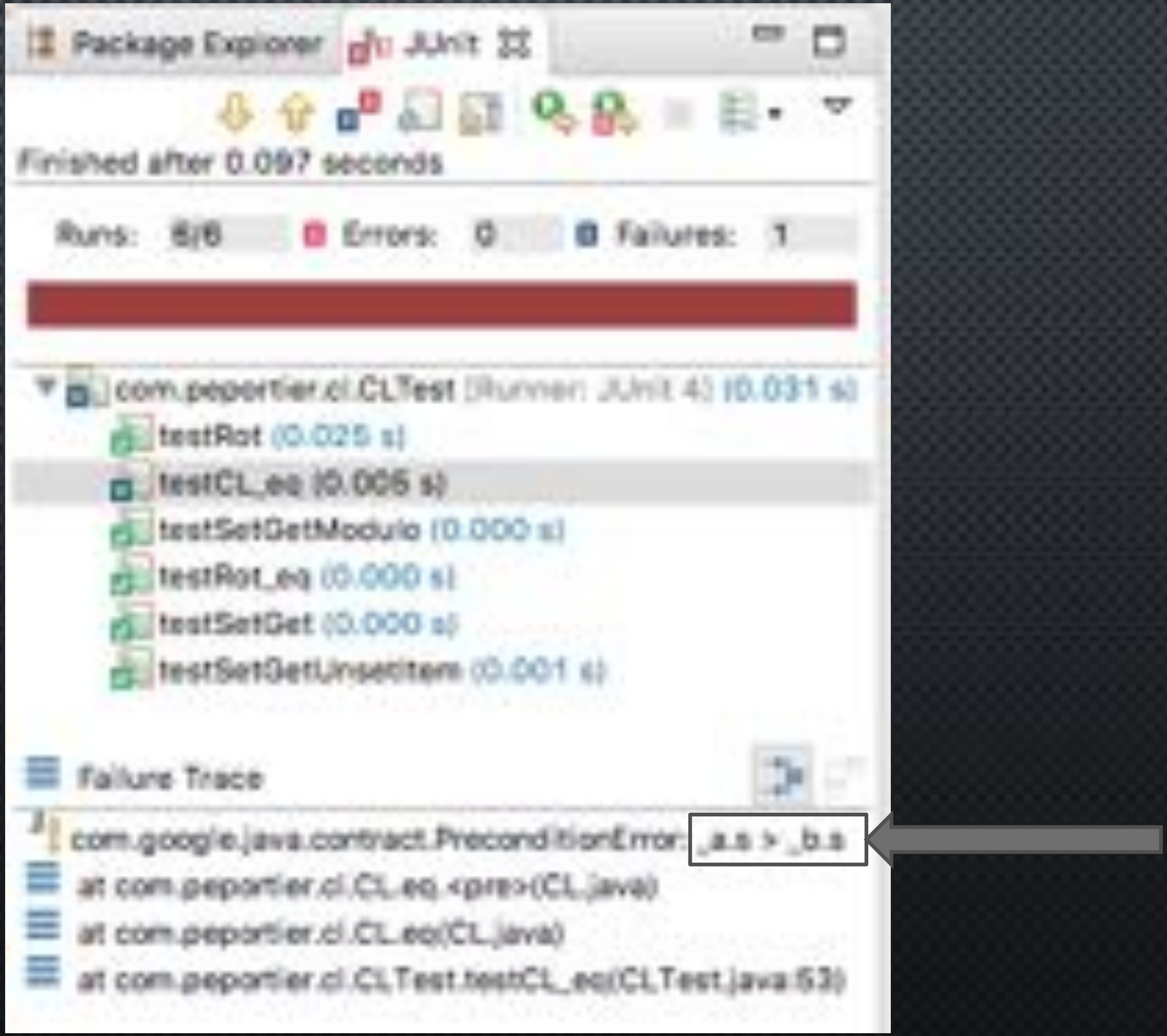
$$R\colon \quad res \equiv \left(\exists i, j :: RA.i = RB.j\right)$$

Naïve solution:
Compare A with each element of RB

```java
@Test
public void testCL_eq() {
    CL a = new CL(3);
     a.set(0,7); a.set(1,9); a.set(2,8);
     CL b = new CL(3);
     b.set(0,9); b.set(1,8); b.set(2,7);
     assertTrue(CL.eq(a, b));
}
```

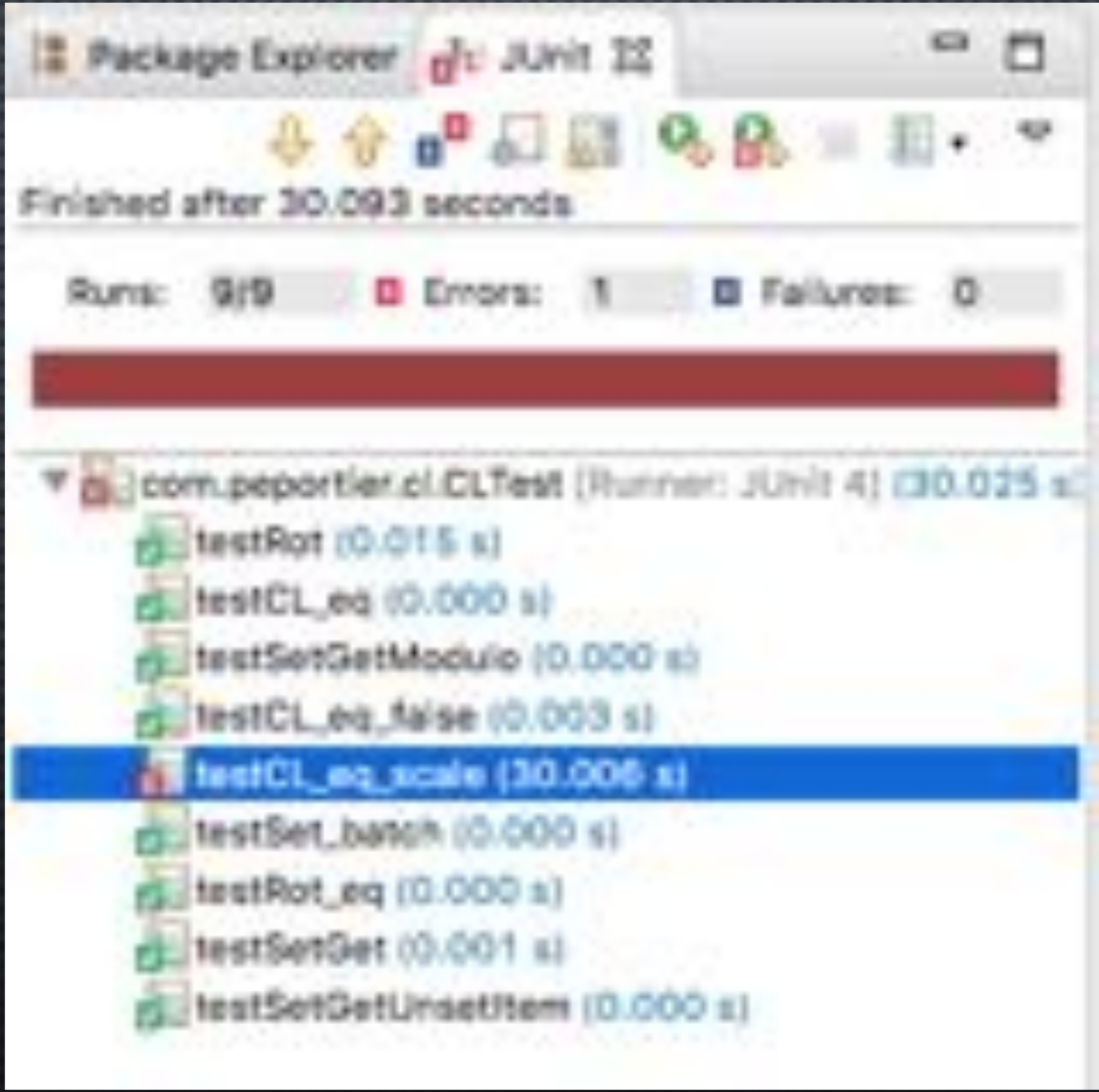
```java
/**
 *
 * @param _a a CL
 * @param _b a CL
 * @return Are the lexicographically sorted versions of a and b are equal?
 */
@Requires("_a.s == _b.s")
public static boolean eq(CL _a, CL _b) {
    for(int i=0 ; i<_a.s ; i++){
        if(Arrays.equals(_a.a, _b.rot(i))) return true;
    }
    return false;
}
```

EXAMPLE OF A COFOJA ERROR



The naïve solution doesn't scale: $O(N^2)$

```java
@Test(timeout=30000)
public void testCL_eq_scale() {
    int SIZE = 100000;
    int[] a_vals = new int[SIZE];
    for (int i = 0 ; i < SIZE ; i++) {
        a_vals[i] = i;
    }
    CL a = new CL(SIZE, a_vals);
    int[] b_vals = a.rot(SIZE/2);
    CL b = new CL(SIZE, b_vals);
    assertTrue(CL.eq(a, b));
}
```


Package Explorer
JUnit
Finished after 30.093 seconds
Runs: 9/9
Errors: 1
Failures: 0
com.peportier.cl.CLTest [Runner: JUnit 4] (30.025 s)
testRot (0.015 s)
testCL_eq (0.000 s)
testSetGetModulo (0.000 s)
testCL_eq_false (0.003 s)
testCL_eq_scale (30.006 s)
testSet_batch (0.000 s)
testRot_eq (0.000 s)
testSetGet (0.001 s)
testSetGetUnsetItem (0.000 s)

RA and RB are either disjoint or equal

$$\big(\forall\, k, i, j :: RA.i \equiv RB.j \Rightarrow RA.(i+k) = RB.(j+k)\big)$$

It is sufficient to compare canonical elements

AA: first element of RA sorted lexicographically

R can be solved by computing AA and BB

To find the solution $res \equiv true$, we can:
(a) Find a pair (i, j) such that $RA.i = RB.j$

To find the solution $res \equiv false$, we can:
(b) Observe $AA \neq BB$

To find the solution $res \equiv true$, we can:

(a) Find a pair (i, j) such that $RA.i = RB.j$

Weakening (a) into a loop invariant:

$$P\colon\ 0 \leq h \quad \wedge \quad (\forall\, k : 0 \leq k < h : RA.i.k = RB.j.k)$$

$$P \wedge h \geq N \quad \Rightarrow \quad true \equiv\ (\exists\, i, j :: RA.i = RB.j)$$

```java
@Test
public void testCL_inv_P() {
    CL a = new CL(3);
    a.set(0,7); a.set(1,9); a.set(2,8);
    CL b = new CL(3);
    b.set(0,9); b.set(1,8); b.set(2,7);
    assertTrue(CL.inv_P(2, 0, 2, a, b));
}
```

```java
@Requires({"_h >= 0", "_i >= 0", "_j >= 0"})
public static boolean inv_P(int _h, int _i, int _j, CL _a, CL _b) {
    boolean res = true;
    for (int k = 0 ; k < _h ; k++)
        res &= _a.rot(_i)[k] == _b.rot(_j)[k];
    return res;
}
```

To find the solution $res \equiv false$, we can:
(b) Observe $AA \neq BB$

Weakening (b) into a loop invariant:

$$QA\text{:}\ \ 0 \leq i \quad \wedge \quad (\forall\, k : 0 \leq k < i : RA.k > BB)$$

$$QA \wedge i \geq N \quad \Rightarrow \quad false \equiv\ (\exists\, i, j :: RA.i = RB.j)$$

Symmetrically for QB
The two CL differ if:

$$QA \wedge QB \wedge (i \geq N\ \vee\ j \geq N)$$

```java
@Test
public void testCL_inv_Q() {
    CL a = new CL(3);
    a.set(0,7); a.set(1,9); a.set(2,8);
    CL b = new CL(3);
    b.set(0,9); b.set(1,8); b.set(2,8);
    assertTrue(CL.inv_Q(2, b, a));
}
```

```java
@Test
public void testCL_smallest() {
    CL a = new CL(3);
    a.set(0,9); a.set(1,8); a.set(2,8);
    int[] expected = {8, 8, 9};
    assertArrayEquals(expected, a.smallest());
}
```

```java
public int[] smallest() {
    int aa[] = a.clone();
    for (int k = 1 ; k < s ; k++) {
        int rot_k[] = rot(k);
        boolean new_smallest = false;
        for (int i = 0 ; i < s ; i++) {
            if         (aa[i] == rot_k[i]) {
                // do nothing
            } else if (aa[i] > rot_k[i]) {
                new_smallest = true; break;
            } else if (aa[i] < rot_k[i]) {
                new_smallest = false; break;
            } else assert false;
        }
        if (new_smallest) aa = rot_k.clone();
    }
    return aa;
}
```

```java
@Requires({"_i >= 0", "_a.s == _b.s"})
public static boolean inv_Q(int _i, CL _a, CL _b) {
    boolean res = true;
    int bb[] = _b.smallest();
    for (int k = 0 ; k < _i ; k++) {
        int rot_k_of_a[] = _a.rot(k);
        for (int j = 0 ; j < _a.s ; j++) {
            if (rot_k_of_a[j] == bb[j]) {
                if (j == (_a.s - 1)) { res = false; break; }
            } else {
                if (rot_k_of_a[j] > bb[j]) {res &= true; break; }
                else { res = false; break; }
            }
        }
        if (res == false) break;
    }
    return res;
}
```

We have the sketch of a solution…

```java
@Requires("_a.s == _b.s")
@Ensures("result == CL.eq(_a,_b)")
public static boolean eq2(CL _a, CL _b) {
    boolean res = false;
    int h,i,j;
    h = i = j = 0;
    assert CL.inv_P(h, i, j, _a, _b);
    assert CL.inv_Q(i, _a, _b);
    assert CL.inv_Q(j, _b, _a);
    while (h<_a.s && i<_a.s && j<_a.s) {
        assert (h+i+j) <= (3*_a.s - 3);
        // increase h+i+j while maintaining P & QA & QB
    }
    assert CL.inv_P(h, i, j, _a, _b);
    assert CL.inv_Q(i, _a, _b);
    assert CL.inv_Q(j, _b, _a);
    assert (h>=_a.s) || (i>=_a.s) || (j>=_a.s);
    if (h >= _a.s) res = true;
    else if (i>=_a.s || j>=_a.s) res = false;
    return res;
}
```

$P\!: \;\; 0 \leq h \quad \wedge \quad (\forall\, k : 0 \leq k < h : RA.i.k = RB.j.k)$

When $RA.i.h = RB.j.h$ (i.e., $A.(i+h) = B.(j+h)$):

$h \leftarrow h+1$ will maintain P and assure progress.

P: $0 \leq h \quad \wedge \quad (\forall\, k : 0 \leq k < h : RA.i.k = RB.j.k)$

QA: $0 \leq i \quad \wedge \quad (\forall\, k : 0 \leq k < i : RA.k > BB)$

$RA.i.h > RB.j.h \quad \wedge \quad P$

$\Rightarrow$ {def. P, $0 \leq p \leq h$}

$RA.i.h > RB.j.h \quad \wedge \quad (\forall\, k : p \leq k < h : RA.i.k = RB.j.k)$

$\Rightarrow$ {lexicographic ordering}

$RA.(i+p) > RB.(j+p)$

$\Rightarrow$ {def. BB}

$RA.(i+p) > BB$

$QA \quad \wedge \quad P \quad \wedge \quad A.(i+h) > B.(j+h)$

$\Rightarrow$ {see above}

$QA \quad \wedge \quad (\forall\, p : 0 \leq p \leq h : RA.(i+p) > BB)$

= {renaming the bounded variable: $i + p = k$}

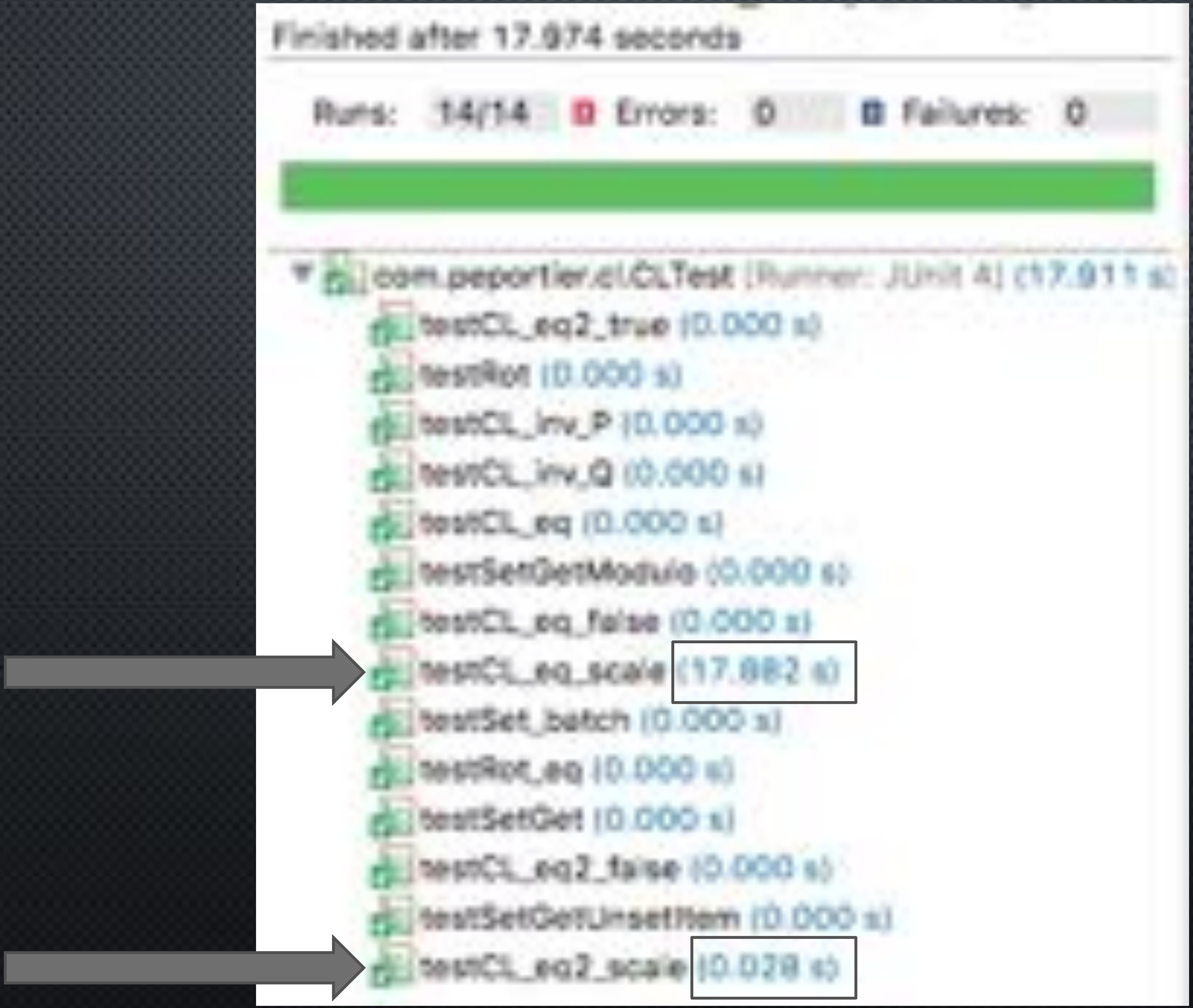
$QA \quad \wedge \quad (\forall\, k : i \leq k \leq i+h : RA.k > BB)$

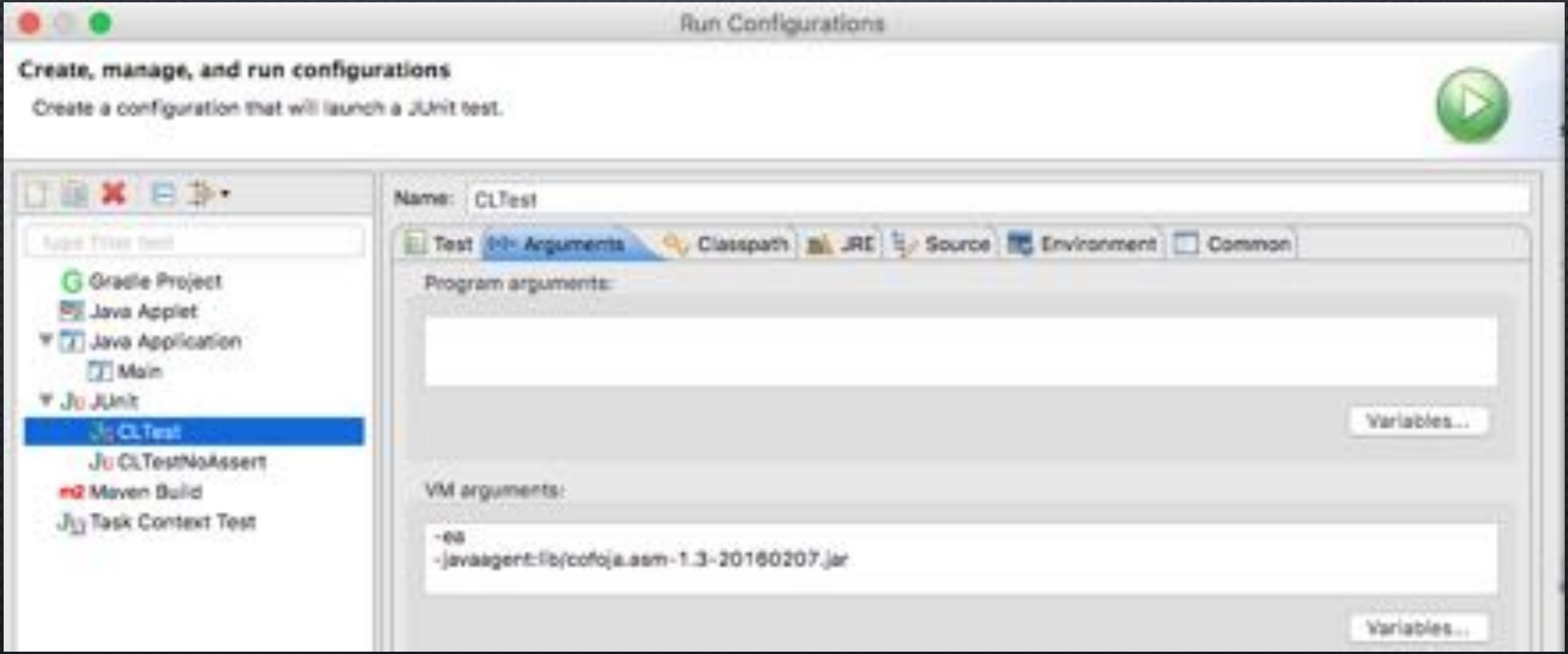
= {def. QA}

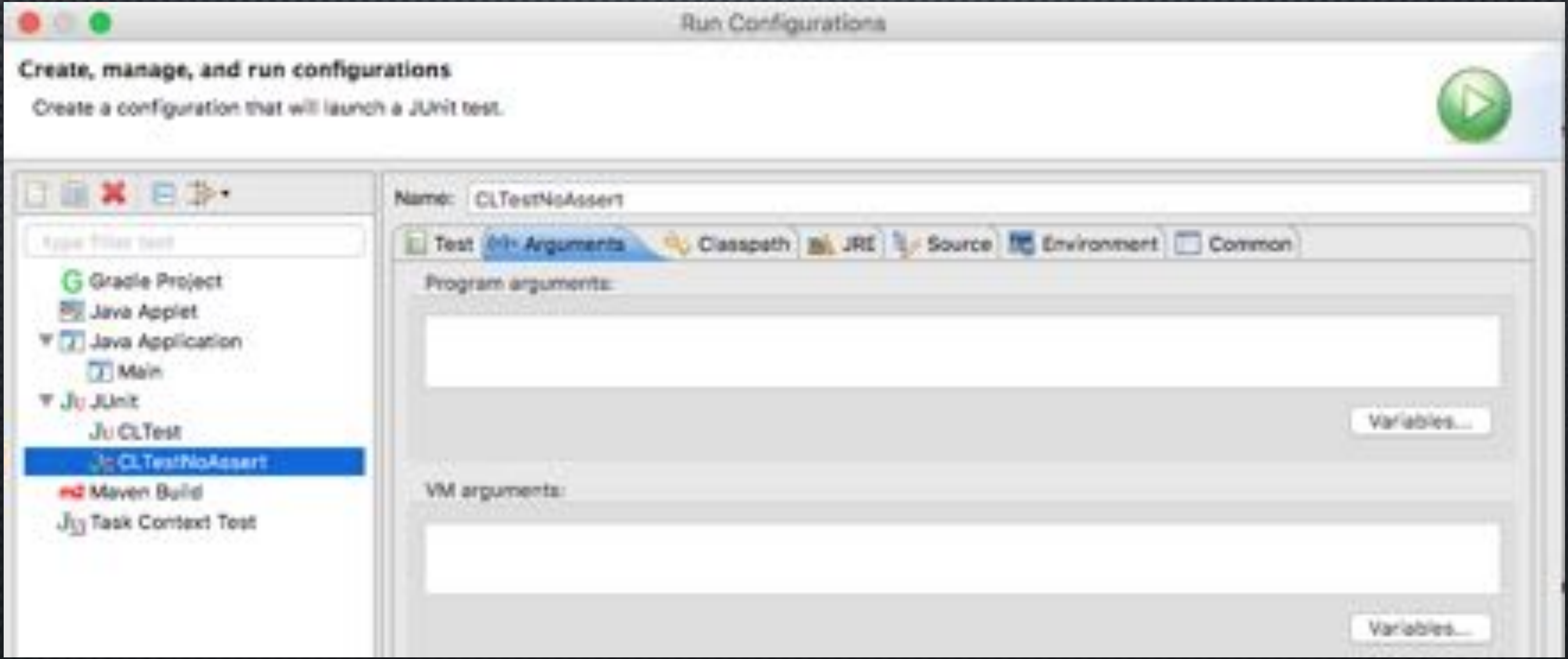
$QA[i \backslash \mathrm{i} + h + 1]$

```
int progress;
while (h<_a.s && i<_a.s && j<_a.s) {
    assert (h+i+j) <= (3*_a.s - 3);
    progress = h+i+j;
    // increase h+i+j while maintaining P & QA & QB
    if      (_a.get(i+h) == _b.get(j+h)) h = h+1;
    else if (_a.get(i+h) > _b.get(j+h)) {i = i+h+1; h=0;}
    else if (_b.get(j+h) > _a.get(i+h)) {j = j+h+1; h=0;}
    else assert false;
    assert CL.inv_P(h, i, j, _a, _b);
    assert CL.inv_Q(i, _a, _b);
    assert CL.inv_Q(j, _b, _a);
    assert h+i+j > progress;
}
```

```java
@Test(timeout=60000)
public void testCL_eq2_scale() {
    int SIZE = 100000;
    int[] a_vals = new int[SIZE];
    for (int i = 0 ; i < SIZE ; i++) {
        a_vals[i] = i;
    }
    CL a = new CL(SIZE, a_vals);
    int[] b_vals = a.rot(SIZE/2);
    CL b = new CL(SIZE, b_vals);
    assertTrue(CL.eq2(a, b));
}
```


Finished after 17.974 seconds
Runs: 14/14
Errors: 0
Failures: 0
com.peportier.cl.CLTest [Runner: JUnit 4] (17.911 s)
testCL_eq2_true (0.000 s)
testRot (0.000 s)
testCL_inv_P (0.000 s)
testCL_inv_Q (0.000 s)
testCL_eq (0.000 s)
testSetGetModule (0.000 s)
testCL_eq_false (0.000 s)
testCL_eq_scale (17.882 s)
testSet_batch (0.000 s)
testRot_eq (0.000 s)
testSetGet (0.000 s)
testCL_eq2_false (0.000 s)
testSetGetUnsetItem (0.000 s)
testCL_eq2_scale (0.028 s)


Run Configurations
Create, manage, and run configurations
Create a configuration that will launch a JUnit test.
Gradle Project
Java Applet
Java Application
Main
JUnit
CLTest
CLTestNoAssert
Maven Build
Task Context Test
Name: CLTest
Test
Arguments
Classpath
JRE
Source
Environment
Common
Program arguments:
Variables...
VM arguments:
-ea
-javaagent:lib/cofoja.asm-1.3-20160207.jar
Variables...


Run Configurations
Create, manage, and run configurations
Create a configuration that will launch a JUnit test.
Name: CLTestNoAssert
Test
Arguments
Classpath
JRE
Source
Environment
Common
Gradle Project
Java Applet
Java Application
Main
JUnit
CLTest
CLTestNoAssert
Maven Build
Task Context Test
Program arguments:
Variables...
VM arguments:
Variables...

http://www.eclemma.org/

```java
@Requires("_a.s == _b.s")
@Ensures("result == CL.eq(_a,_b)")
public static boolean eq2(CL _a, CL _b) {
    boolean res = false;
    int h,i,j;
    h = i = j = 0;
    assert CL.inv_P(h, i, j, _a, _b);
    assert CL.inv_Q(i, _a, _b);
    assert CL.inv_Q(j, _b, _a);
    while (h<_a.s && i<_a.s && j<_a.s) {
        assert (h+i+j) <= (3*_a.s - 3);
        // increase h+i+j while maintaining P & QA & QB
        if        (_a.get(i+h) == _b.get(j+h)) h = h+1;
        else if (_a.get(i+h) > _b.get(j+h)) {i = i+h+1; h=0;}
        else if (_b.get(j+h) > _a.get(i+h)) {j = j+h+1; h=0;}
        else assert false;
    }
    assert CL.inv_P(h, i,
    assert CL.inv_Q(i, _a, _b);
    assert CL.inv_Q(j, _b, _a);
    assert (h>=_a.s) || (i>=_a.s) || (j>=_a.s);
    if (h >= _a.s) res = true;
    else if (i>=_a.s || j>=_a.s) res = false;
    return res;
}
```

1 of 2 branches missed.

```java
@Requires({"_i >= 0", "_a.s == _b.s"})
public static boolean inv_Q(int _i, CL _a, CL _b) {
    boolean res = true;
    int bb[] = _b.get_copy_arr();
    Arrays.sort(bb);
    for (int k = 0 ; k < _i ; k++) {
        int rot_k_of_a[] = _a.rot(k);
        for (int j = 0 ; j < _a.s ; j++) {
            if (rot_k_of_a[j] == bb[j]) {
                if (j == (_a.s - 1)) { res = false; break; }
            } else {
                if (rot_k_of_a[j] > bb[j]) {res &= true; break; }
                else { res = false; break; }
            }
        }
        if (res == false) break;
    }
    return res;
}
```

Shiloach, Yossi.
"A fast equivalence-checking algorithm for circular lists."
Information Processing Letters 8.5 (1979): 236-238.

Gasteren, Antonetta JM.
"On the shape of mathematical arguments. "
Vol. 445. *Springer Science & Business Media*, 1990.

https://en.wikipedia.org/wiki/Lexicographically_minimal_string_rotation

LINKS

- [HTTP://DOCS.ORACLE.COM/JAVASE/6/DOCS/TECHNOTES/GUIDES/LANGUAGE/ASSERT.HTML #USAGE-CONDITIONS]
- [HTTPS://GITHUB.COM/NHATMINHLE/COFOJA]
- [HTTPS://WWW.EIFFEL.COM/VALUES/DESIGN-BY-CONTRACT/]
- [HTTPS://GITHUB.COM/JUNIT-TEAM/JUNIT4/WIKI/ASSERTIONS]
- [HTTP://WWW.VOGELLA.COM/TUTORIALS/JUNIT/ARTICLE.HTML#USING-JUNIT-INTEGRATED-INTO-ECLIPSE]