Homework 1

Below are four faulty programs. Each includes test inputs that result in failure. Answer the following questions about each program.

```
/**
* Find last index of element
*
* @param x array to search
* @param y value to look for
* @return last index of y in x; -1 if absent
* @throws NullPointerException if x is null
*/
public int findLast (int[] x, int y)
{
   for (int i=x.length-1; i > 0; i--)
   {
        if (x[i] == y)
        {
             return i;
        }
   }
   return -1;
}
// test: x = [2, 3, 5]; y = 2; Expected = 0
// Book website: FindLast.java
// Book website: FindLastTest.java
```

```
/**
* Find last index of zero
*
* @param x array to search
*
* @return last index of 0 in x; -1 if absent
* @throws NullPointerException if x is null
*/
public static int lastZero (int[] x)
{
   for (int i = 0; i < x.length; i++)
   {
        if (x[i] == 0)
        {
             return i;
        }
    }
    return -1;
}
// test: x = [0, 1, 0]; Expected = 2
// Book website: LastZero.java
// Book website: LastZeroTest.java
```

```
/**
* Count positive elements
*
* @param x array to search
* @return count of positive elements in x
* @throws NullPointerException if x is null
*/
public int countPositive (int[] x)
{
    int count = 0;
    for (int i=0; i < x.length; i++)
    {
        if (x[i] >= 0)
        {
           count++;
        }
    }
    return count;
}
// test: x = [-4, 2, 0, 2]; Expcted = 2
// Book website: CountPositive.java
// Book website: CountPositiveTest.java
```

```
/**
* Count odd or postive elements
*
* @param x array to search
* @return count of odd/positive values in x
* @throws NullPointerException if x is null
*/
public static int oddOrPos(int[] x)
{
    int count = 0;
    for (int i = 0; i < x.length; i++)
    {
        if (x[i]%2 == 1 || x[i] > 0)
        {
            count++;
        }
     }
     return count;
}
// test: x = [-3, -2, 0, 1, 4]; Expected = 3
// Book website: OddOrPos.java
// Book website: OddOrPosTest.java
```

(a) Explain what is wrong with the given code. Describe the fault precisely by proposing a modification to the code.

	wrong part &modification	explanation
findLast	for (int i=x.length-1; i > 0; i--) ⇨ for (int i=x.length-1; i ≥ 0; i--)	應涵蓋 index 0
lastZero	for (int i = 0; i < x.length; i++) ⇨ for (int i = x.length-1 ; i ≥0 ; i--)	應從 x[] 後找至前
countPositive	if (x[i] >= 0) {count++} ⇨ if (x[i] > 0) {count++}	要求正數，不是非負數
oddOrPos	if (x[i]%2 == 1 || x[i] > 0) ⇨ if (x[i]%2 !=0 || x[i] > 0)	x[i]%2 = ±1 未涵蓋負奇數

(b) If possible, give a test case that does not execute the fault. If not, briefly explain why not.

	Test case	Expected / Result	Reason
findLast	x = []; y = 1;	-1	NullPointerException before the loop test is evaluated.
lastZero	必會執行 fault，因為 fault 出在迴圈本身		
countPositive	x = [];	0	NullPointerException before the loop test is evaluated.
oddOrPos	x = [];	0	NullPointerException before the loop test is evaluated.

(c) If possible, give a test case that executes the fault, but does not result in an error state. If not, briefly explain why not.

	Test case	Expected	Result	reason
findLast	x = [1, 2]; y = 2;	1	1	最後一個 y 不在 index = 0 處
lastZero	Impossible，因為必存在 error state。 考慮以下程式： for (int i = 0; i < x.length; i++) ⇨ for (int i = x.length-1 ; i ≥0 ; i--) --- 不論 x =[] 或含有大於 0 個元素，index i 都存在 error state (原程式的 i > 改良版程式的 i)，故index i 必有 error state			
countPositive	x = [1, 2];	2	2	不含0
oddOrPos	x = [1, 2]	2	2	不含負奇數

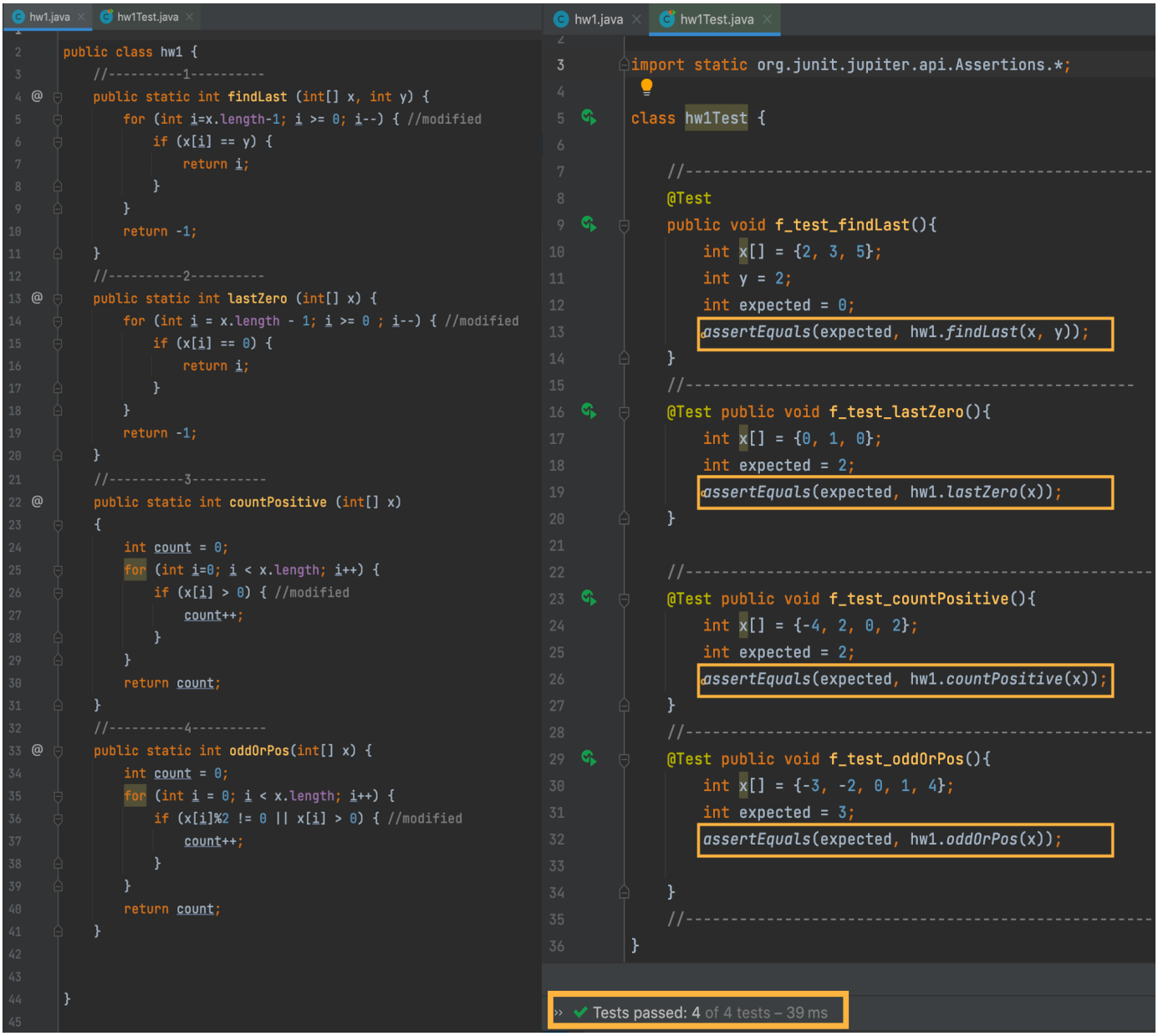
(d) If possible, give a test case that results in an error state, but not a failure. Hint: Don't forget about the program counter. If not, briefly explain why not.

	Test case	Expected	Result	reason
findLast	x = [1, 2]; y = 2;	1	1	index i 有 error state
lastZero	x =[1, 0];	1	1	index i 有 error state
countPositive	Impossible，一旦 count 出錯(出現 0 使 count 多加一次)，之後的 count 都會不同。			
oddOrPos	Impossible，一旦 count 出錯(出現負奇數使 count 少加一次)，之後 count 都會不同。			

(e) For the given test case, describe the first error state. Be sure to describe the complete state.

	Test case	Expected	First error state
findLast	x = [2, 3, 5]; y = 2;	0	i = 0 PC = just before return -1
lastZero	x = [0, 1, 0];	2	i = 0 PC = just after i = 0
countPositive	x = [-4, 2, 0, 2]	2	i = 2 count = 1 PC = immediately before the count++ statement.
oddOrPos	x = [-3, -2, 0, 1, 4]	3	i = 0 count = 0 PC = at end of if statement, instead of just before count++

(f) Implement your repair and verify that the given test now produces the expected output. Submit a screen printout or other evidence that your new program works.



hw1.java

```
public class hw1 {
    //----------1----------
    public static int findLast (int[] x, int y) {
        for (int i=x.length-1; i >= 0; i--) { //modified
            if (x[i] == y) {
                return i;
            }
        }
        return -1;
    }
    //----------2----------
    public static int lastZero (int[] x) {
        for (int i = x.length - 1; i >= 0 ; i--) { //modified
            if (x[i] == 0) {
                return i;
            }
        }
        return -1;
    }
    //----------3----------
    public static int countPositive (int[] x)
    {
        int count = 0;
        for (int i=0; i < x.length; i++) {
            if (x[i] > 0) { //modified
                count++;
            }
        }
        return count;
    }
    //----------4----------
    public static int oddOrPos(int[] x) {
        int count = 0;
        for (int i = 0; i < x.length; i++) {
            if (x[i]%2 != 0 || x[i] > 0) { //modified
                count++;
            }
        }
        return count;
    }


}
```

hw1Test.java

```
import static org.junit.jupiter.api.Assertions.*;

class hw1Test {

    //-----------------------------------------------
    @Test
    public void f_test_findLast(){
        int x[] = {2, 3, 5};
        int y = 2;
        int expected = 0;
        assertEquals(expected, hw1.findLast(x, y));
    }
    //-----------------------------------------------
    @Test public void f_test_lastZero(){
        int x[] = {0, 1, 0};
        int expected = 2;
        assertEquals(expected, hw1.lastZero(x));
    }

    //-----------------------------------------------
    @Test public void f_test_countPositive(){
        int x[] = {-4, 2, 0, 2};
        int expected = 2;
        assertEquals(expected, hw1.countPositive(x));
    }
    //-----------------------------------------------
    @Test public void f_test_oddOrPos(){
        int x[] = {-3, -2, 0, 1, 4};
        int expected = 3;
        assertEquals(expected, hw1.oddOrPos(x));

    }
    //-----------------------------------------------
}
```

Tests passed: 4 of 4 tests – 39 ms