Chapter 1: Why do we test?

1. **Chapter 1, exercise #5**

   Four faulty programs are given below. Each includes test inputs that result in failure. For each of these four programs, answer the following questions **a — f**. For **"f"** copy and paste your corrected code in this document.

1.1. `findLast` function

```
/**
 * 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
```

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

The i>0 condition in the for loop is wrong because it doesn't check the first index of the array. It should be i>=0.

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

x = null

y = 2

Expected Output = NullPointerException

Actual Output = NullPointerException

By passing a null argument into the method for parameter x, the program will throw a null pointer exception during the initialization condition of the for loop, so the fault will not be reached.

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.

x = [2,3,5]

y = 5

Expected Output = 2

Actual Output = 2

State-0	x = [2,3,5]	y = 5	i = undefined	PC = findLast()
State-1	x = [2,3,5]	y = 5	i = undefined	PC = before i=x.length-1
State-2	x = [2,3,5]	y = 5	i = 2	PC = after i=x.length-1
State-3	x = [2,3,5]	y = 5	i = 2	PC = i>0;
State-4	x = [2,3,5]	y = 5	i = 2	PC = if(x[i]==y);
State-5	x = [2,3,5]	y = 5	i = 2	PC = return i

The fault line is executed at State-3, none of these states are incorrect, so there is no error. As long as a match exists in the array in index 1 or greater, then the faulty testing condition will be reached, but will not cause the error state, because the code will return the correct answer.

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

x = [2,3,5]

y = 9

Expected Output = -1

Actual Output = -1

State-0	x = [2,3,5]	y = 9	i = undefined	PC = findLast()
State-1	x = [2,3,5]	y = 9	i = undefined	PC = before i=x.length-1
State-2	x = [2,3,5]	y = 9	i = 2	PC = after i=x.length-1
State-3	x = [2,3,5]	y = 9	i = 2	PC = i>0;
State-4	x = [2,3,5]	y = 9	i = 2	PC = if(x[i]==y);
State-5	x = [2,3,5]	y = 9	i = 1	PC = i--;
State-6	x = [2,3,5]	y = 9	i = 1	PC = i>0;
State-7	x = [2,3,5]	y = 9	i = 1	PC = if(x[i]==y);
State-8	x = [2,3,5]	y = 9	i = 0	PC = i--;

State-9 x = [2,3,5] y = 9 i = 0 PC = i>0;

State-10 x = [2,3,5] y = 9 i = 0(undefined) PC = return -1;

State-10 is an incorrect state causing an error. However, the output of the program matches with the expected output. The method fails to check the first element in the array, normally, i=2,1,0 but in our case i=2,1 causing an error but not a failure.

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

x = [2,3,5] y = 2

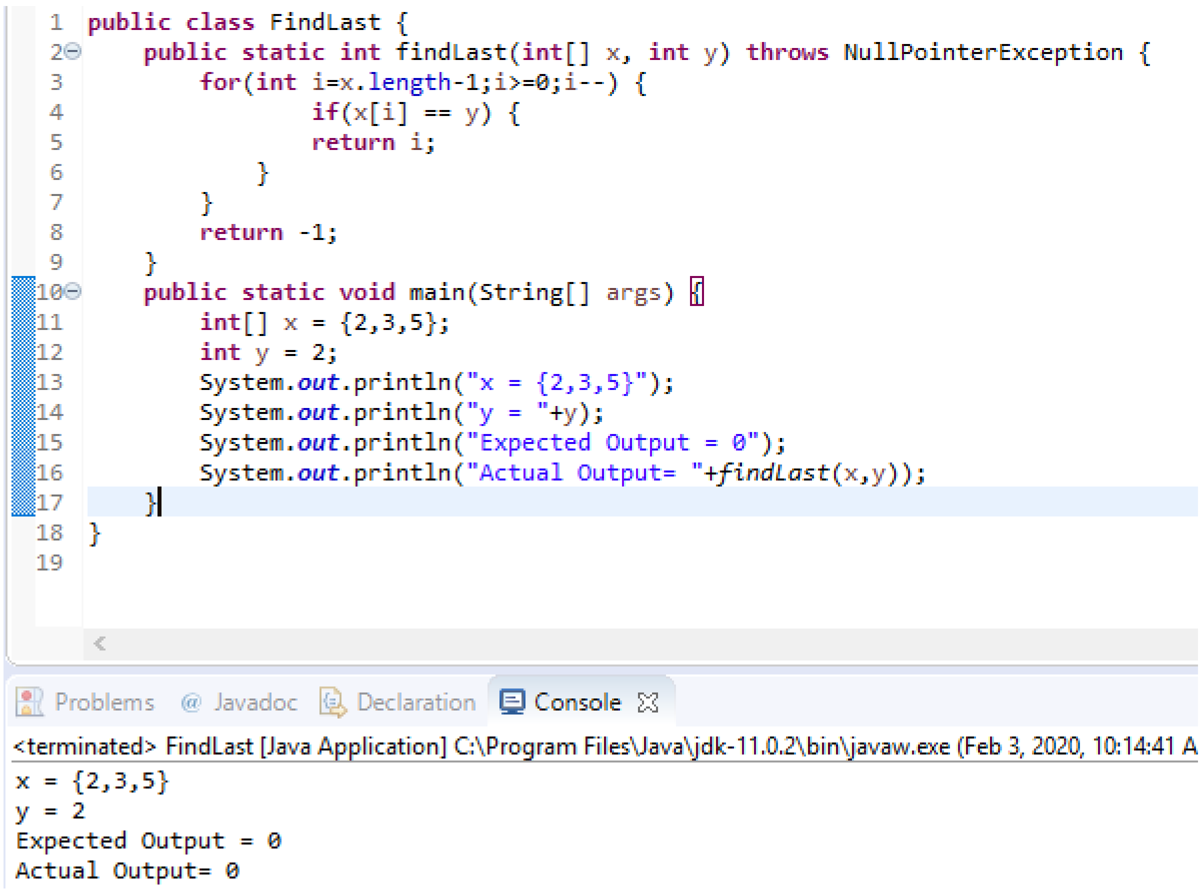
Expected Output: 0

Actual Output: -1

First error state:

x = [2,3,5] y=2 i=0 (or undefined) PC = just before return -1;

f) Implement your repair and verify that the given test now produces the expected output. Submit a screenshot demonstrating your new program works.



```java
public class FindLast {
    public static int findLast(int[] x, int y) throws NullPointerException {
        for(int i=x.length-1;i>=0;i--) {
                if(x[i] == y) {
                return i;
            }
        }
        return -1;
    }
    public static void main(String[] args) {
        int[] x = {2,3,5};
        int y = 2;
        System.out.println("x = {2,3,5}");
        System.out.println("y = "+y);
        System.out.println("Expected Output = 0");
        System.out.println("Actual Output= "+findLast(x,y));
    }
}
```

```
<terminated> FindLast [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe (Feb 3, 2020, 10:14:41 A
x = {2,3,5}
y = 2
Expected Output = 0
Actual Output= 0
```

1.2. `lastZero` function

```
/**
 * 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
```

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

The code given the for loop starts from beginning of the array and returns when it sees zero.

Instead for loop should start from last index of the array.

The for loop should start from high to low.

for(int i=x.length-1; i>=0;i--)

This will return the last zero.

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

All inputs start the loop, so all inputs execute the fault even the null input.

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.**

If the loop is not unrolled at all, there is no error.

All inputs (except x= [0]) result in an error state. This is because if the array has one entry which is zero, i never takes the value -1.

x = [7] will result in an error state because in the correct program i will be decremented to -1 before returning whereas in the incorrect version i incremented to 1 before the return. The only exception is x = [0] in which case i gets the exact same value in both correct and incorrect versions.

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

There is an error every time the loop is unrolled more than once, since the values of index i ascend instead of descend.

x = [2,0,5]

Expected Output: 1

Actual Output: 1

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

x = [0,1,0]

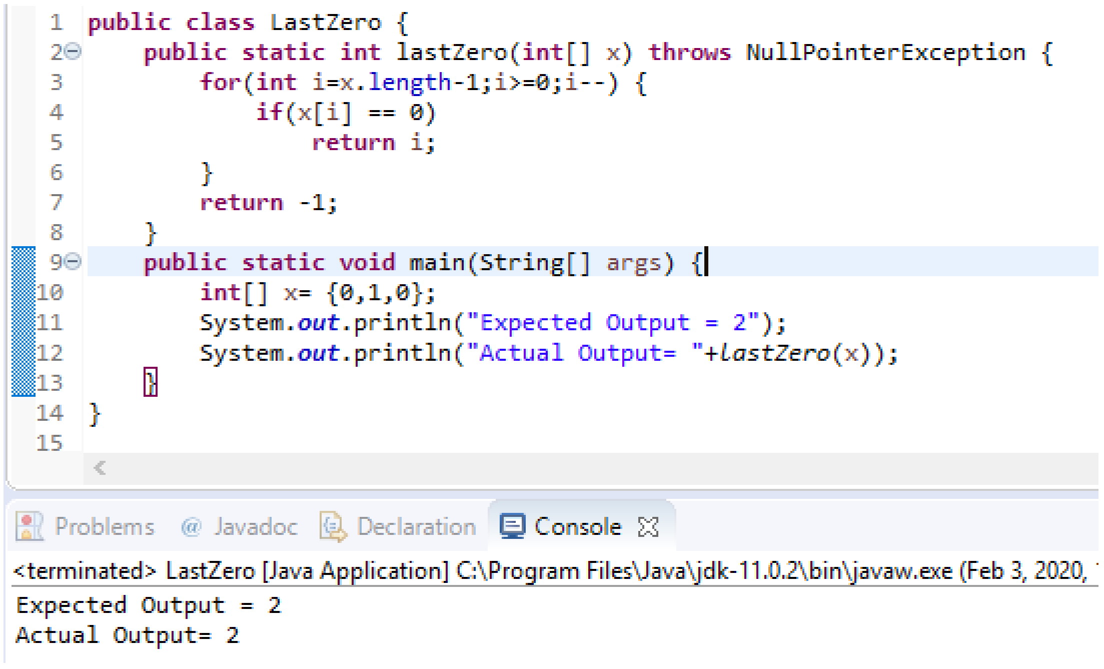
Expected Output: 2

Actual Output: 0

First error state:

x = [0,1,0]   i = 0   PC = just after i = 0;

f) Implement your repair and verify that the given test now produces the expected output. Submit a screenshot demonstrating your new program works.



```java
public class LastZero {
    public static int lastZero(int[] x) throws NullPointerException {
        for(int i=x.length-1;i>=0;i--) {
            if(x[i] == 0)
                return i;
        }
        return -1;
    }
    public static void main(String[] args) {
        int[] x= {0,1,0};
        System.out.println("Expected Output = 2");
        System.out.println("Actual Output= "+lastZero(x));
    }
}
```

```
<terminated> LastZero [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe (Feb 3, 2020,
Expected Output = 2
Actual Output= 2
```

1.3. `countPositive` function

```
/**
 * 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
```

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

Zero is neither negative nor positive. So, the if condition if(x[i] >=0) is wrong because it considers zero as positive. The correct if condition should be if(x[i]>0).

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

If x is null or zero, the fault will not be executed.

If x is null, it will give a NullPointerException.

If x is zero, it will not go into the for loop and return count = 0.

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.

x = [-4,1,2,3]

Expected Output: 3

Actual Output: 3

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

Every input that results in error also results in failure. The reason is that error states are not repairable by subsequent processing. If there is a 0 in x, all subsequent states will be error states no matter what else is in x.

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

x = [-4,2,0,2]

Expected Output: 2

Actual Output: 3

First error state:

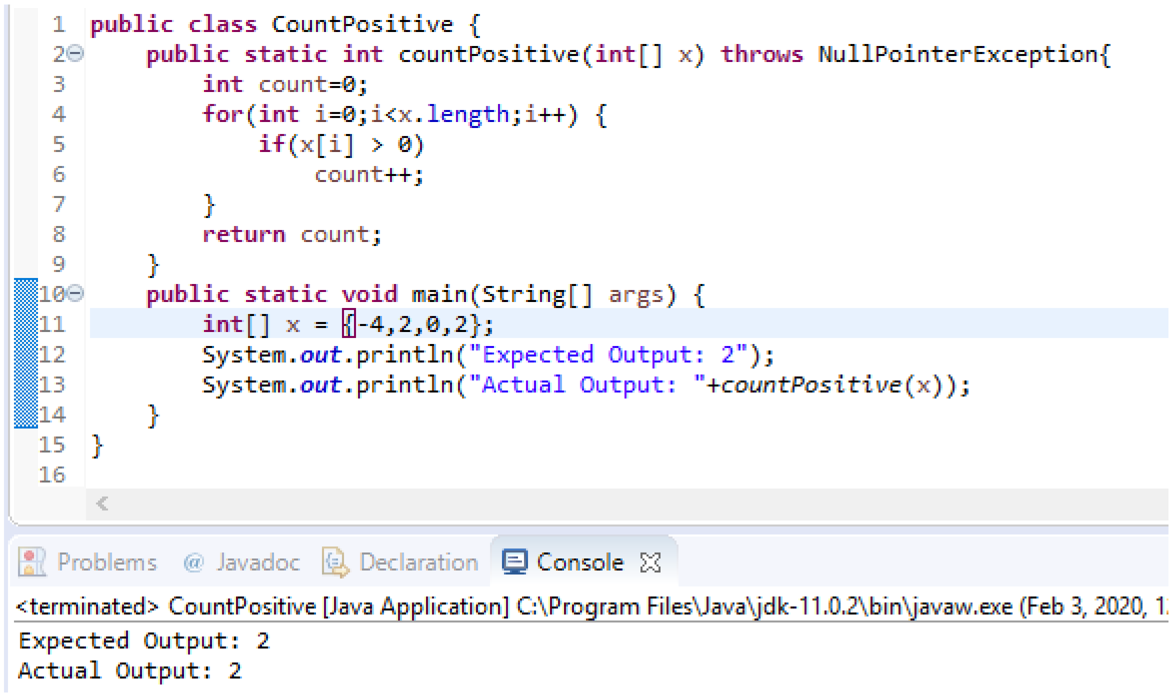
x = [-4,2,0,2]

i=2

count=1

PC = immediately after the count++ statement.

f) Implement your repair and verify that the given test now produces the expected output. Submit a screenshot demonstrating your new program works.



```java
public class CountPositive {
    public static int countPositive(int[] x) throws NullPointerException{
        int count=0;
        for(int i=0;i<x.length;i++) {
            if(x[i] > 0)
                count++;
        }
        return count;
    }
    public static void main(String[] args) {
        int[] x = {-4,2,0,2};
        System.out.println("Expected Output: 2");
        System.out.println("Actual Output: "+countPositive(x));
    }
}
```

```
<terminated> CountPositive [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe (Feb 3, 2020, 1
Expected Output: 2
Actual Output: 2
```

1.4. `oddOrPos` function

```
/**
 * 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.

The if-test needs to take account of negative values (positive odd numbers are taken care of by the second test x[i]>0).So, first test condition should be changed to x[i]%2==-1. This will cover all negative odd numbers and x[i] >0 will cover all positive numbers including odds.

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

x must be either null or empty. All other inputs result in the fault being executed.

x = []

Expected Output = 0

Actual Output = 0

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.

Any nonempty x with only non-negative elements works because the first part of the compound if-test is not necessary unless the value is negative.

x = [3,4,5]

Expected Output = 3

Actual Output = 3

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

For this program, every input that results in error also results in failure. The reason is that error states are not repairable by subsequent processing. If there is a negative value in x, all subsequent states will be error states no matter what else is in x.

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

x = [-3,-2,0,1,4]

Expected Output = 3

Actual Output = 2

First Error State

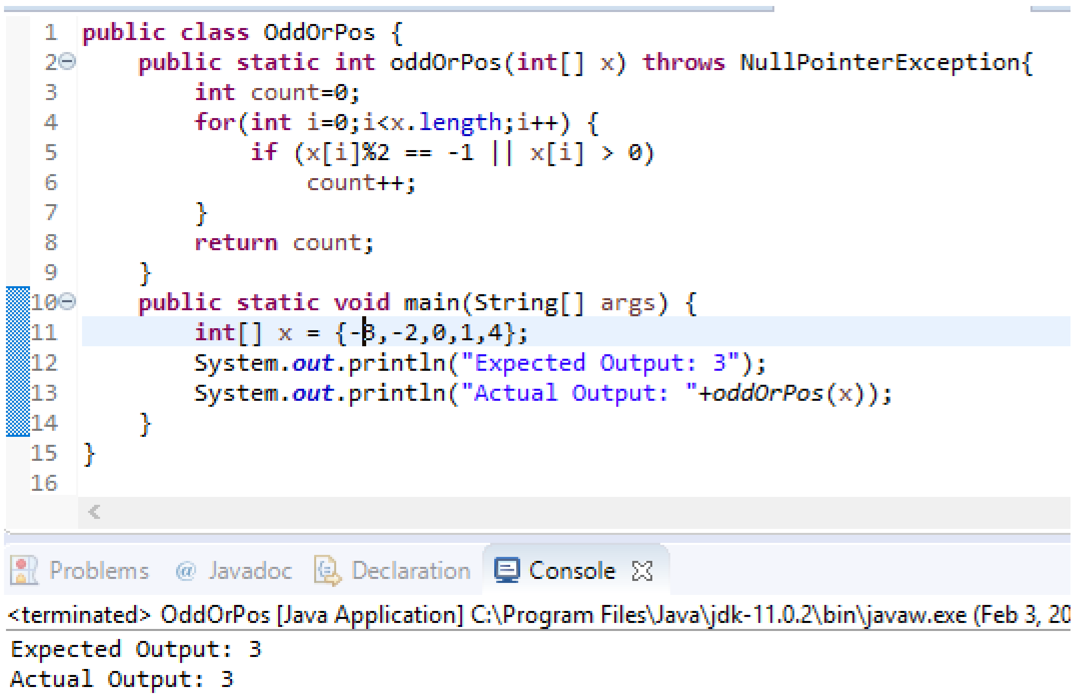
x = [-3,-2,0,1,4]

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 screenshot demonstrating your new program works.



```java
public class OddOrPos {
    public static int oddOrPos(int[] x) throws NullPointerException{
        int count=0;
        for(int i=0;i<x.length;i++) {
            if (x[i]%2 == -1 || x[i] > 0)
                count++;
        }
        return count;
    }
    public static void main(String[] args) {
        int[] x = {-3,-2,0,1,4};
        System.out.println("Expected Output: 3");
        System.out.println("Actual Output: "+oddOrPos(x));
    }
}
```

```
<terminated> OddOrPos [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe (Feb 3, 20
Expected Output: 3
Actual Output: 3
```