Daniel Tam
1001724986

Program 1:

[a] - The problem is that the program only checks if the index is greater than 0 instead of greater than or equal to 0. This will result in the program never checking the final index in the given array. Changing the check to greater than or equal to 0 will fix the problem: for(int i = x.length-1; **i >= 0**; i--)

[b] - A test case that will not execute the fault would be if x = null. If the array is null, it will throw a NullPointerException and not execute the fault.

[c] - A test case that executes the fault but does not result in an error state would be if x = [2, 3, 5] and y = 3, as it will be able to find the correct value even if it runs through a faulty statement

[d] - A test case that results in an error but not a failure would be if x = [2, 3, 5] and y = 4. There is an error since it never checks the initial index, but since 4 is not in our array, it would not result in a failure since we are not expecting it to find the number 4.

[e] - The first error state would be when i = 0. Since we keep subtracting from i in our loop, whenever it reaches 0, it would produce the first error state since it will check against if i is greater than 0 and since i is 0, it will incorrectly skip over the first item in the array as that is index 0.

Input: x = [2, 3, 5], y = 2
Expected output: 0
Actual output: -1

First state
i = 2
x[i] = 5
y = 2

Second state
i = 1
x[i] = 3
y = 2

Third state (First error state - skipped)
(expected)
i = 0

[f] - (Below)

Program 2:

[a] - The problem with the code is that it is counting 0 as a positive element. If we change x[i] >= 0 to x[i] > 0, it should no longer count 0 as a positive element.

[b] - A test case that would not execute the fault would be if x = null. This will throw a NullPointerException and would not check against the fault.

[c] - A test case that executes the fault but does not result in an error state would be if x = [-4, 2, 1, 2]. Since we never check against 0, it will display the correct value.

Daniel Tam
1001724986

[d] - There is no test case in which it can result in an error but not a failure. Since the error is the program claiming that 0 is a positive integer, the error will always result in a failure.

[e] - The first error state would occur when it firsts checks that it is greater than or equal to 0. Which should be in the third iteration of the loop. It finds a 0 as the third item in the array, and counts it as positive thus adding it to our counter.

Input: x = [-4, 2, 0, 2]
Expected output: 2
Actual output: 3

First state
i = 0
x[i] = -4
count = 0

Second state
i = 1
x[i] = 2
count = 1

Third state (First error state)
i = 2
x[i] = 0
count = 2

Fourth state
i = 3
x[i] = 2
count = 3

[f] - (Below)

Daniel Tam
1001724986

```java
public class assignment1
{
	public static int findLast (int[] x, int y)
	{
		for(int i=x.length-1; i >= 0; i--)
		{
			if(x[i] == y)
			{
				return i;
			}
		}

		return -1;
	}

	public static int countPositive(int[] x)
	{
		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[] = {2, 3, 5};
		int y = 2;
		System.out.printf("findLast%nExpected: 0%nResult: %d%n%n", findLast(x, y));

		int z[] = {-4, 2, 0, 2};
		System.out.printf("countPositive%nExpected: 2%nResult: %d%n", countPositive(z));
	}
}
```

```
daniel@Daniel-Main ~/D/S/C/assignment1> java assignment1
findLast
Expected: 0
Result: 0

countPositive
Expected: 2
Result:2
daniel@Daniel-Main ~/D/S/C/assignment1>
```