**Assignment 5**

**Testing and Debugging**

**Name:**

*Below are four faulty programs. Each includes a test case that results in failure. Answer the following questions about each program.*

public int findLast (int[] x, int y)

{

// Effects: If x==null throw NullPointerException

// else return the index of the last element

// in x that equals y.

// If no such element exists, return -1

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

{

if (x[i] == y)

{

return i;

}

}

return -1;

}

// Test input: x=[2, 3, 5]; y = 2

// Expected output = 0

1. Identify the fault.
2. If possible, identify a test case that does not execute the fault (provide justification).
3. If possible, identify a test case that executes the fault, but does not result in an error state.
4. Fix the fault and verify that the given test now produces the expected output.

public static int lastZero (int[] x)

{

//Effects: if x==null throw NullPointerException

// else return the index of the LAST 0 in x.

// Return -1 if 0 does not occur in x

for (int i = 0; i < x.length; i++)

{

if (x[i] == 0)

{

return i;

}

}

return -1;

}

// Test input: x=[0, 1, 0]

// Expected output = 2

1. Identify the fault.
2. If possible, identify a test case that does not execute the fault (provide justification).
3. If possible, identify a test case that executes the fault, but does not result in an error state.
4. Fix the fault and verify that the given test now produces the expected output.

public int countPositive (int[] x)

{

//Effects: If x==null throw NullPointerException

// else return the number of

// positive (non-zero) elements in x.

int count = 0;

for (int i=0; i < x.length; i++)

{

if (x[i] >= 0)

{

count++;

}

}

return count;

}

// Test input: x=[-4, 2, 0, 2]

// Expected output = 2

1. Identify the fault.
2. If possible, identify a test case that does not execute the fault (provide justification).
3. If possible, identify a test case that executes the fault, but does not result in an error state.
4. Fix the fault and verify that the given test now produces the expected output.

public static int oddOrPos(int[] x)

{

//Effects: if x==null throw NullPointerException

// else return the number of elements in x that

// are either odd or positive (or both)

int count = 0;

for (int i = 0; i < x.length; i++)

{

if (x[i]%2 == 1 || x[i] > 0)

{

count++;

}

}

return count;

}

// Test input: x=[-3, -2, 0, 1, 4]

// Expected output = 3

1. Identify the fault.
2. If possible, identify a test case that does not execute the fault.
3. If possible, identify a test case that executes the fault, but does not result in an error state.
4. Fix the fault and verify that the given test now produces the expected output.