Name: Natnael Kebede

ID: 1001149004

3. 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:  x=[2, 3, 5]; y = 2
  //        Expected = 0
```

```
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:  x=[0, 1, 0]
  //        Expected = 2
```

```
public int countPositive (int[] x) {
//Effects: If x==null throw NullPointerException
//   else return the number of
//        positive elements in 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]
  //        Expected = 2
```

```
public static int odd0rPos(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:  x=[-3, -2, 0, 1, 4]
  //        Expected = 3
```

(a) Identify the fault.
(b) If possible, identify a test case that does **not** execute the fault.
(c) If possible, identify a test case that executes the fault, but does **not** result in an error state.
(d) If possible identify a test case that results in an error, but **not** a failure. Hint: Don't forget about the program counter.
(e) For the given test case, identify the first error state. Be sure to describe the complete state.
(f) Fix the fault and verify that the given test now produces the expected output.

1 a) the fault in the **findLast** method is that the first element of the test case or 2 is never checked.

b) If we consider a test case where x = null and y = 2 the fault won't be executed.

c) If we consider a test case where x = [2, 3, 5] and y = 5 the fault will be executed but it won't result in an error state.

d) If we consider a test case where x = [2, 3, 5] and y = 0 it results in an error but not in a failure.

e) The first error state for the given test case is x = [2, 3, 5], y = 2, x.length = 3, i = 0 and PC = return - 1

f) To fix the fault we have to change the condition i > 0 to i >= 0. This will result in the expected output for the test case given in the question.

2 a) the fault in the **LastZero** method is that the program returns the index of the first 0 in x but not the last zero in x.

b) If we consider a test case where x = null the fault won't be executed.

c) If we consider a test case where x = [0] the fault will be executed but it won't result in an error state.

d) If we consider a test case where x = [0, 1, 1] it results in an error but not in a failure.

e) The first error state for the given test case is x = [0, 1, 0], x.length = 3, i = 0, PC = just after i = 0

f) To fix the fault we have to change the for loop to be for (int i = x.length -1; i >= 0; i--) This will result in the expected output for the test case given in the question.

3 a) the fault in the **countPositive** method is that the count includes 0 but zero is not positive in x.

b) If we consider a test case where x = null the fault won't be executed.

c) If we consider a test case where x = [2] the fault will be executed but it won't result in an error state.

d) It seems impossible to find a test case that results in an error but not in a failure. That is, every input that results in error also results in failure.

e) The first error state for the given test case is x = [-4, 2, 0, 2], x.length = 4, i = 2, count = 0,

PC = if

f) To fix the fault we have to change the if statement inside the for loop to be if (x[i] > 0). This will result in the expected output for the test case given in the question.

4 a) the fault in the **oddOrPos** method is that the code doesn't consider non-negative odd numbers.

b) If we consider a test case where x = null the fault won't be executed.

c) If we consider a test case where x = [3, 2, 1] the fault will be executed but it won't result in an error state.

d) It seems impossible to find a test case that results in an error but not in a failure. That is, every input that results in error also results in failure.

e) The first error state for the given test case is x = [-3, -2, 0, 1, 4], x.length = 5, i = 0, count = 0, PC = if

f) To fix the fault we have to change the if statement inside the for loop to be if (x[i]%2 == 1 || x[i] > 0 || x[i]%2 == -1). This will result in the expected output for the test case given in the question.