Exercício REO 2: Defeito, Erro e Falha

Abaixo são apresentados quatro programas com defeito. Para cada programa é ilustrado um caso de teste que resulta em uma falha. Responda as questões abaixo.

----------Programa 1----------

```
public int findLast(int[] x, int y){
//effects: if x == null throw
NullPointerException
//else return the index of the last
element
//int 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
// expect = 0
```

----------Programa 2----------

```
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]
// expect = 2
```

----------Programa 3----------

```
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]
// expect = 2
```

----------Programa 4----------

```
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: x=[-3,-2,0,1,4]
// expect = 3
```

1. Identifique o defeito de cada programa.
2. Se possível, identifique um caso de teste que não executa o defeito. Explique.
3. Se possível, identifique um caso de teste que executa o defeito, mas não resulta em um erro. Explique.
4. Se possível, identifique um caso de teste que resulta em um erro, mas não em uma falha. Explique.
5. Corrija o defeito e verifique se o caso de teste dado produza saída esperada.